Visual-Inertial Curve SLAM: Creating a Sparse Structured World without Feature Points

Kevin Meier
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
kcmeier2@illinois.edu

Soon-Jo Chung
Associate Professor of Aerospace and Bren Scholar
California Institute of Technology
1200 E California Blvd, MC 105-50
Pasadena, CA 91125
sjchung@caltech.edu

Seth Hutchinson
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
seth@illinois.edu

Abstract

We present a simultaneous localization and mapping (SLAM) algorithm that uses Bézier curves as static landmark primitives rather than feature points. Our approach allows us to estimate the full 6-DOF pose of a robot while providing a structured map which can be used to assist a robot in motion planning and control. We demonstrate how to reconstruct the 3-D location of curve landmarks from a stereo pair and how to compare the 3-D shape of curve landmarks between chronologically sequential stereo frames to solve the data association problem. We also present a method to combine curve landmarks for mapping purposes, resulting in a map with a continuous set of curves that contain fewer landmark states than conventional point-based SLAM algorithms. We demonstrate our algorithm's effectiveness with numerous experiments, including comparisons to existing state-of-the-art SLAM algorithms.

1 Introduction

As SLAM has matured as a discipline, SLAM research has increasingly focused on systems-level issues such as optimizing constituent components of algorithms and overall systems integration [Zhang et al., 2015, Mur-Artal et al., 2015, Lu and Song, 2015]. While in the early days of SLAM, researchers often presented results of robot excursions measured in meters, today, systems are expected to perform well over much longer trajectories [Li and Mourikis, 2013, Luetenegger et al., 2015, Mur-Artal et al., 2015, Jones and Soatto, 2011], further motivating the emphasis of a systems-level approach. At the forefront of the vision-based SLAM approach, feature points are usually selected to represent landmarks in the map. Although point-based approaches have produced precise SLAM systems that run in real-time, point-based SLAM algorithms are subject to a number of drawbacks: Points ignore structural information between sampling points belonging to the same surface or edge, making it difficult for a robot to determine how it should interact with its surroundings. Creating dense maps with feature points requires a large state space. Many map points do not represent anything physically significant and are not needed because they belong to a structured object

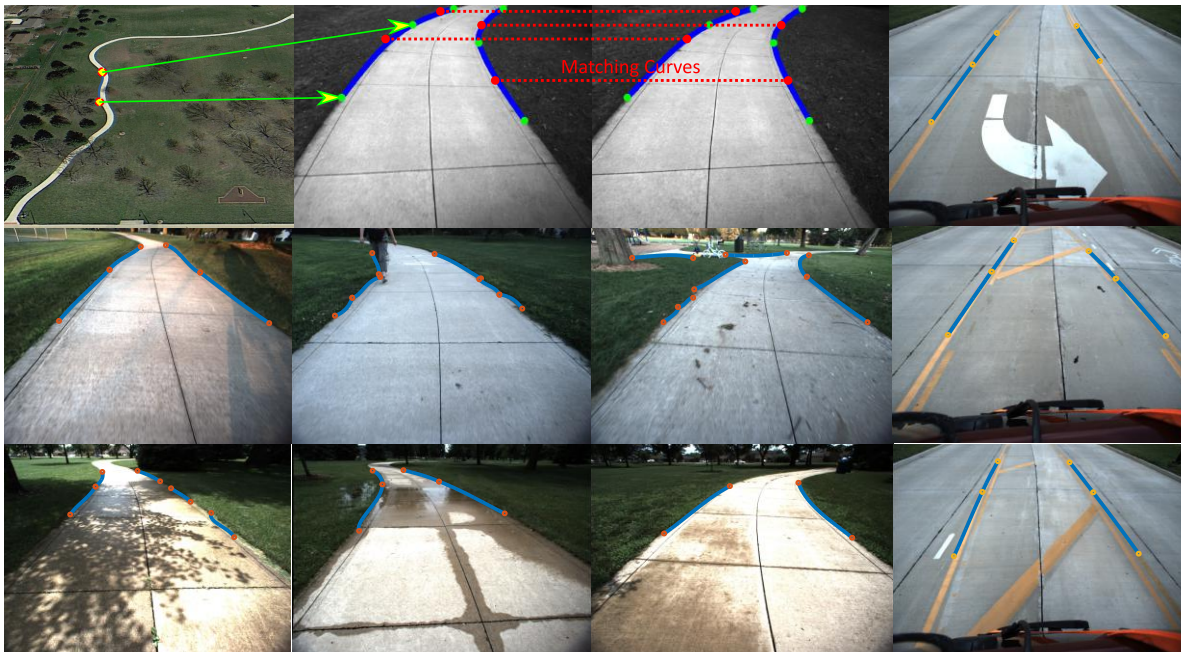


Figure 1: The images demonstrate the proposed Curve SLAM algorithm applied to various settings under different environmental conditions. Curve SLAM relies on a stereo camera and IMU to solve the SLAM problem in a previously unknown environment using parameterized curves as landmarks. The images show curve landmarks interpolated to yellow road lanes and the outline of a sidewalk.

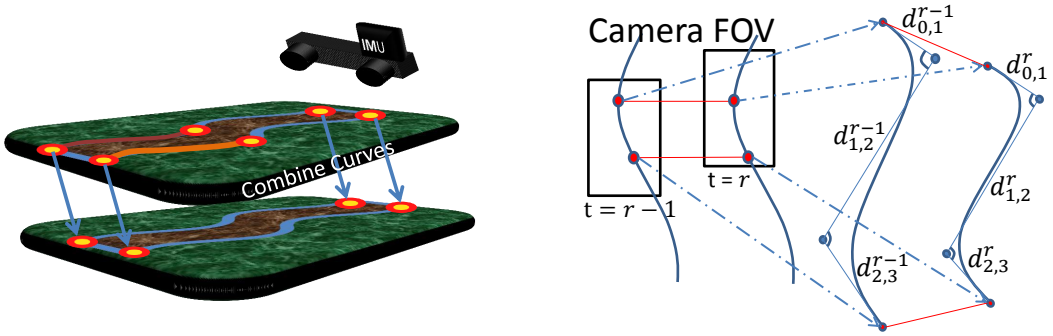
that can be represented compactly using parameterized shapes. In settings that lack distinguishable feature points, point-based detector/descriptor algorithms will fail to track enough feature points for a robot to accurately estimate its pose. In contrast, our proposed Curve SLAM algorithm is able to operate in settings that lack distinguishable feature points while creating sparse stuctured maps of the environment. In fact, in our experimental evaluations, we have observed that Curve SLAM can reduce the required number of landmark features by several orders of magnitude relative to state-of-the-art point-based methods.

In this article, we present a systems-level approach that uses Bézier curves as landmarks in the SLAM framework as opposed to feature points. Our work derives its motivation from environments where a river, road, or path dominates the scene. In these environments, distinctive feature points may be scarce. As shown in Figure 1, we overcome these problems by exploiting the structure of the road, river, or path, using Bézier curves to represent the edges of the path. Then, with a stereo camera and IMU, we reconstruct the 3-D location of these curves while simultaneously estimating the 6-DOF pose of a robot.

1.1 Contributions

This paper presents a much improved version of our previous Curve SLAM approaches [Rao et al., 2012] [Meier et al., 2016]. Our contribution is a systems-level approach that extends and combines methods in computer vision and computer-aided geometric design to solve the SLAM problem. The specific contributions and benefits of the proposed approach can be summarized as follows:

- We present an algorithm that interpolates, splits, and matches curves in a stereo pair. This allows us to reconstruct the 3-D location of curves, parameterized by a small number of control points.
- We present a data association algorithm that compares the physical dimensions of curve landmarks between chronologically sequential stereo image frames to remove curve outliers, see Figure 2. When



(a) Combining Curves.

(b) Comparing Curves.

Figure 2: Features of the proposed Curve SLAM algorithm. As shown in Figure 2a, the proposed algorithm provides a method that combines curves to reduce the data representing the map. Figure 2a demonstrates a before (top layer) and after (bottom layer) effect of our curve combining algorithm. Curve SLAM also provides a method to compare the physical dimensions of curve landmarks (see Figure 2b), allowing Curve SLAM to operate in settings that lack distinguishable feature points. Figure 2b is further explained in Section 4.1.

the dimensions of a curve do not match between sequential image frames, the curve is pruned. The algorithm relies on heuristics and mild assumptions, but is designed to find false associations when a small number of landmark curves (usually less than four) are consistently tracked between image frames. Tracking such a small number of landmarks allows our algorithm to operate in settings that lack distinguishable feature points, and to create maps that are more sparse than point-based SLAM algorithms. Our approach to the data association problem is quite different from point-based algorithms that track hundreds of feature points between image frames.

- We present a curve combining algorithm that reduces the number of curve landmarks representing the map, see Figure 2. Bézier curves are useful in this process because they can be represented compactly by the location of parameterized control points, allowing us to construct large maps with fewer landmark states than conventional point-based SLAM algorithms. Additionally, the shape of curves provides structure and information which can aid a robot in path planning and control.

- We validate the approach with experimental hardware in multiple outdoor environments, and we compare Curve SLAM against the Open Keyframe-based Visual-Inertial SLAM (OKVIS) [Lueteneg-ger et al., 2015] algorithm and a stereo version of the Parallel Tracking and Mapping (SPTAM) algorithm [Pire et al., 2015] [Klein and Murray, 2007].

1.2 Related Work

Various SLAM algorithms have incorporated high-level features in order to overcome drawbacks associated with point-based SLAM. Examples of high-level features include planes [Nguyen et al., 2007] [Gee et al., 2008] [Ozog et al., 2015], image moments [Dani et al., 2013], line segments [Zhang et al., 2015] [Smith et al., 2006] [Lu and Song, 2015], objects such as office chairs and tables [Salas-Moreno et al., 2013], or a river [Nuske et al., 2015] [Yang et al., 2015]. A desirable characteristic of high-level structure is that it provides a compact structured map of the environment. For instance, in [Nguyen et al., 2007] orthogonal planes are used to represent structured landmarks in a compact manner. The orthogonal planes represent objects such as walls, the ceiling, windows, and doors of an indoor office setting. In a similar fashion, the work in [Smith et al., 2006] and [Zhang et al., 2015] use line segments to localize a camera, and to map an environment with a vision-based sensor. Lines represent the structure of objects one would expect to find in the mapped location, e.g., a computer monitor, the structure of an indoor hallway, or the outline of a door. Lines also provide a sparse representation of the environment. For example, in [Smith et al., 2006], only two points,

the start and end point of a line segment, are used to represent each line. The work in [Lu and Song, 2015] presents a systems-level SLAM approach that uses line segments, ideal lines, vanishing points, and primary planes that are used in conjunction with feature points. In addition to providing a structured map, their algorithm demonstrates that high-level features can improve the localization accuracy of a SLAM algorithm when used in conjunction with sparse feature points, and they demonstrate that high-level features are able to operate in settings that lack distinguishable feature points. Later in this article, we demonstrate that Curve SLAM shares this characteristic of being able to operate in settings that lack conspicuous feature points.

By creating compact structured maps of the environment, the Curve SLAM algorithm incorporates some of the ideas represented in the previously mentioned papers on high-level structure, and the systems-level approach we take is similar in form to the recent publications [Lu and Song, 2015] [Zhang et al., 2015]. However, Curve SLAM is different from the previously mentioned algorithms on high-level structure because Curve SLAM is intended for settings that contain curved features, e.g., settings where a path or road is present. Applying Bézier curves as landmark primitives in these settings allows us create maps that are more sparse than the high-level feature primitives previously mentioned.

The use of curves as vision primitives has been studied in the computer vision literature. Methods have been devised to match curves across multiple image views [Schmid and Zisserman, 2000], not necessarily closely spaced or specifically in stereo images [Kedem and Yarmovski, 1996]. The work in [Xiao and Li, 2005] presents a method to reconstruct the 3-D location of non-uniform rational B-spline curves from a single stereo pair without matching point-based stereo correspondences. We incorporate the idea contained in this paper in Section 3.

The algorithm in [Pedraza et al., 2009] uses B-spline curves as landmark primitives to localize a robot and create a structured map of an environment in a compact fashion. They use a single plane 2-D scanning laser sensor and an extended Kalman filter (EKF) to jointly estimate the 3-DOF pose of a planar ground robot and the location of each B-spline curve. They demonstrate how to add curves to their filter state and extend the length of curves in their filter state by modifying the location of parameterized curve control points representing each B-spline curve. Our work differs from the work in [Pedraza et al., 2009] in a number of ways. The algorithms in this article are designed around a vision sensor as its primary sensing input as opposed to a 2-D scanning laser. Thus most of the constituent algorithms in this paper, such as reconstructing 3-D curves or solving the data association step, require an entirely different approach. Additionally, our state is more general than the filter state in [Pedraza et al., 2009]; we include the full 6-DOF pose of a robot, making it applicable to various robotic platforms such as small UAVs. Furthermore, because one of our sensing inputs is a stereo camera, we are able to locate curves in settings where a laser sensor will fail, e.g., in one of our experiments, we use yellow road lanes as curve landmarks.

Various place recognition algorithms have been developed that can aid a SLAM system that is occasionally unable to track feature points. A recent and thorough review of the place recognition problem is given in [Lowry et al., 2016]). Unfortunately, most approaches rely on feature points or likely require offline training. Additionally, feature points will likely fail when the appearance of the scene changes drastically, e.g., unexpected weather, or changes in lighting conditions [Furgale and Barfoot, 2010]. The work in [Paton et al., 2016] presents an algorithm that is invariant to lighting conditions, but relies on feature points and requires multiple stereo cameras. The work in [McManus et al., 2015, Linegar et al., 2016] presents a place recognition algorithm that is invariant to seasonal and lighting changes, and does not require feature points. Their work uses mid-level image patches, and a support vector machine to train an image classifier offline. They demonstrate that their algorithm is invariant to extreme changes in the environment. Unfortunately, their work requires at least one pass of the environment and offline training.

Before proceeding, we emphasize that we do not believe that point-based approaches are necessarily inferior. Indeed, recent point-based SLAM approaches demonstrate remarkably accurate results that run in real-time [Kaess et al., 2012, Keivan and Sibley, 2015, Mourikis and Roumeliotis, 2007, Li and Mourikis, 2013, Jones and Soatto, 2011, Kelly and Sukhatme, 2011, Konolige and Agrawal, 2008, Kim et al., 2015, Klein and Murray,

2007]; this is true both in estimating the motion of a vehicle and in creating maps of an environment. In fact, Curve SLAM can be modified rather easily to include feature points so that curves and feature points can be used simultaneously to solve the SLAM problem. However, we believe alternative approaches are required in order to overcome the aforementioned shortcomings inherent with feature points.

2 Curve SLAM Overview

2.1 Goals and Assumptions

The aim of this paper is to estimate the pose of a robot equipped with a stereo camera and IMU while providing a sparse structured map of a previously unknown environment using static curved features as landmarks. Letting $\mathbf{x} \in \mathbb{R}^{9+12N}$ represent the state vector, our goal is to estimate the following variables:

$$\mathbf{x} \triangleq [\mathbf{p}^{\mathcal{W}}, \mathbf{v}^{\mathcal{B}}, \boldsymbol{\Theta}, (\mathbf{C}_1^{\mathcal{W}})^\top, \ldots, (\mathbf{C}_N^{\mathcal{W}})^\top]^\top \tag{1}$$

where $\mathbf{p}^{\mathcal{W}}$ represents the robot's position with respect to the world frame, $\mathbf{v}^{\mathcal{B}}$ represents the body-frame velocity of the robot, and $\boldsymbol{\Theta}$ represents the robot's attitude. The N variables $\mathbf{C}_1^{\mathcal{W}}, \ldots, \mathbf{C}_N^{\mathcal{W}}$ represent the location of curved features defined with respect to the world-frame. Each curved feature $\mathbf{C}_j^{\mathcal{W}} \in \{\mathbf{C}_1^{\mathcal{W}}, \ldots, \mathbf{C}_N^{\mathcal{W}}\}$ is represented as a Bézier curve and is defined by the location of its control points, i.e., $\mathbf{C}_j^{\mathcal{W}} \triangleq [\,(\mathbf{P}_{j,0}^{\mathcal{W}})^\top,\ (\mathbf{P}_{j,1}^{\mathcal{W}})^\top,\ (\mathbf{P}_{j,2}^{\mathcal{W}})^\top,\ (\mathbf{P}_{j,3}^{\mathcal{W}})^\top]^\top \in \mathbb{R}^{12}$ where the variables $\mathbf{P}_{j,0}^{\mathcal{W}} \in \mathbb{R}^3$, $\mathbf{P}_{j,1}^{\mathcal{W}} \in \mathbb{R}^3$, $\mathbf{P}_{j,2}^{\mathcal{W}} \in \mathbb{R}^3$, and $\mathbf{P}_{j,3}^{\mathcal{W}} \in \mathbb{R}^3$ represent the 3-D coordinates of control points defined with respect to the world frame (an image of a cubic Bézier curve, along with its control points, is shown in Figure 3). Each curved feature is fixed to a larger curved object naturally occurring in the scene, e.g., a long curved sidewalk. Additionally, because the work in this paper is focused on mapping environments where a road or path dominates the scene, we assume at least one static curve is in the image corresponding to the left or right edge of a road or path.

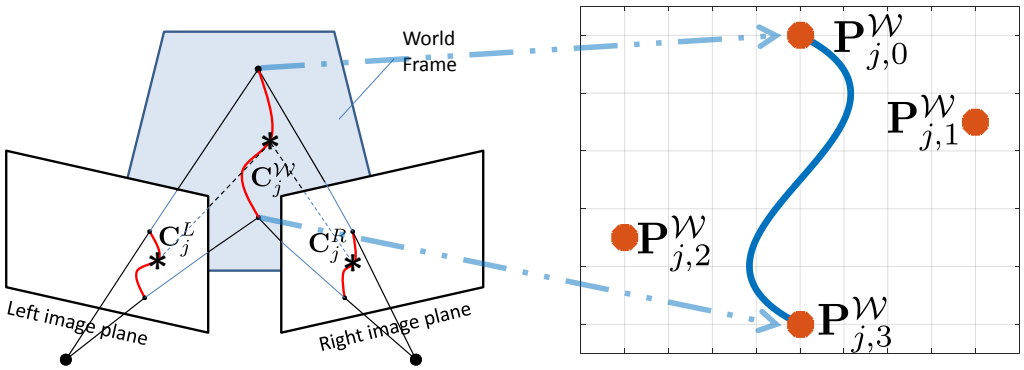


Figure 3: The goal of Curve SLAM is to simultaneously estimate the pose of a robot and the 3-D location of curved features. Each curved feature is represented as a Bézier curve and is defined by the location of its 3-D control points: $\mathbf{P}_{j,0}^{\mathcal{W}}$, $\mathbf{P}_{j,1}^{\mathcal{W}}$, $\mathbf{P}_{j,2}^{\mathcal{W}}$, and $\mathbf{P}_{j,3}^{\mathcal{W}}$.

2.2 Outline of Curve SLAM Algorithm

The pseudocode of the proposed Curve SLAM algorithm is provided in Algorithm 1, and the functional components of Curve SLAM are illustrated in Figure 4. As shall be demonstrated in Section 3, the algorithm uses a single stereo pair to determine the 3-D coordinates of curve landmark parameters, i.e., control points, relative to the body frame of the stereo camera. To reconstruct the 3-D location of control points, we do

not rely on matching point-to-point stereo correspondences. Instead, we use the projection of curves in the stereo image plane to formulate a least-squares problem that optimizes the 3-D location of control points. Section 4 explains how to track curve landmarks between chronologically sequential image frames in order to solve the data association problem. Section 4 also explains how the IMU measurements and the control point measurements captured from the stereo camera are fused together with an EKF to simultaneously localize the stereo camera and create a structured map. Once the location of a 3-D curve has been estimated, Section 4.8 shows how to combine this curve with previously estimated curves to further reduce the number of curve landmarks representing the map.

Algorithm 1 Curve SLAM

1: Initialize the camera's pose and breakpoints (Sections 4.2 and 4.3).
2: **while** A new stereo image pair is available **do**
3: Extract the boundary of the path in the left and right stereo pair (Section 3.1).
4: With the breakpoints, interpolate and match curves between the left and right stereo pair (Section 3.2).
5: With the Levenberg-Marquardt algorithm, reconstruct the 3-D location of curve control points (Section 3.3).
6: Track curves between the current and previous frame (Section 4.1).
7: Verify that physical dimensions of curves match between the current and previous frame. Remove outliers as necessary (Section 4.1).
8: If necessary, add curves in the image plane, and determine the curve's polynomial order. (Sections 4.2 and 4.3).
9: Assign breakpoints as the start and end control points of the tracked curves from step 6, and newly added curves from step 7.
10: Compute the EKF prediction using the IMU (Section 4.5).
11: Compute the EKF update with the reconstructed control points obtained in step 4 (Section 4.6).
12: If necessary, add newly found image curves from step 7 to the filter state (Section 4.7).
13: When a curve leaves the camera's field of view, combine it with previously estimated curves (Section 4.8).
14: **end while**

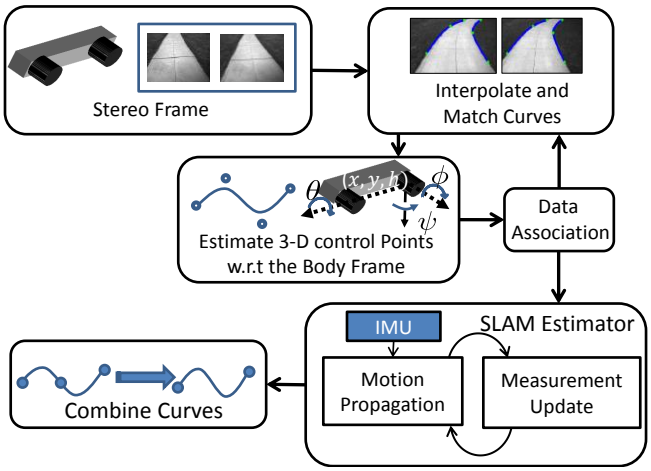


Figure 4: Functional components of the Curve SLAM algorithm.

2.3 Properties of Bézier Curves

The notations and symbols used throughout the paper are defined in Table I. The following properties of Bézier curves [Piegl and Tiller, 1997] make them useful for Curve SLAM:

- A Bézier curve is defined by its control points, which define the curve's shape. The start control point and end control point are located at the start point and end point of the curve. Letting t_i be the i^{th} element of $\mathbf{t} = [0, \Delta t, 2\Delta t, \ldots, 1]^\top$, the linear transformation that maps t_i to a point that lies on the Bézier curve $\mathbf{C}_j^{\mathcal{W}}$ is given by

Table 1: Notations and Definitions

Name	Description
$\mathcal{W}$	The 3-D world frame.
$\mathcal{B}$	The body frame of the stereo camera. When a subscript is attached to the variable, e.g., $\mathcal{B}_r$, it is used to denote the body frame at the r^{th} stereo image frame.
L	The left camera image.
R	The right camera image.
$\mathbf{p}^{\mathcal{W}}$	$\mathbf{p}^{\mathcal{W}} \triangleq (x, y, z)$ is defined as the 3-D displacement of $\mathcal{B}$ with respect to $\mathcal{W}$. The variable $h = -z$ represents the camera's height.
$\mathbf{\Theta}$	$\mathbf{\Theta} \triangleq (\phi, \theta, \psi)$ is the orientation of $\mathcal{B}$ in ZYX euler angles.
$\mathbf{v}^{\mathcal{B}}$	$\mathbf{v}^{\mathcal{B}} \triangleq (u, v, w)$ is defined as the linear velocity of the camera with respect to the body frame.
$\mathbf{T}_{AB}$	$\mathbf{T}_{AB} \in SE(3)$ is the transformation that changes the representation of a point defined in the coordinates of frame B to a point defined in the coordinates of frame A.
$\mathbf{P}_{j,l}^{\mathcal{W}} \in \mathbb{R}^3$	We define the variables $\mathbf{P}_{j,0}^{\mathcal{W}}$, $\mathbf{P}_{j,1}^{\mathcal{W}}$, $\mathbf{P}_{j,2}^{\mathcal{W}}$, $\mathbf{P}_{j,3}^{\mathcal{W}}$ to represent the control points of the j^{th} cubic Bézier curve. The variables $\mathbf{P}_{j,1}^{\mathcal{W}}$ and $\mathbf{P}_{j,2}^{\mathcal{W}}$ are the middle control points, and $\mathbf{P}_{j,0}^{\mathcal{W}}$ and $\mathbf{P}_{j,3}^{\mathcal{W}}$ are the start and end control points, respectively. The superscript denotes the frame where the variable is defined. When the curve is linear or quadratic, $\mathbf{P}_{j,0}^{\mathcal{W}}$ and $\mathbf{P}_{j,3}^{\mathcal{W}}$ are the start and end control points, respectively. When the curve is quadratic, $\mathbf{P}_{j,1}^{\mathcal{W}}$ is the middle control point.
$\mathcal{O}_j$	$\mathcal{O}_j$ denotes the order of the j^{th} curve. In this article, $\mathcal{O}_j$ is 1 (linear), 2 (quadratic), or 3 (cubic).
$\mathbf{C}_j^{\mathcal{W}}$	$\mathbf{C}_j^{\mathcal{W}} \triangleq [\ (\mathbf{P}_{j,0}^{\mathcal{W}})^\top,\ (\mathbf{P}_{j,1}^{\mathcal{W}})^\top,\ (\mathbf{P}_{j,2}^{\mathcal{W}})^\top,\ (\mathbf{P}_{j,3}^{\mathcal{W}})^\top]^\top \in \mathbb{R}^{12}$ is defined as the j^{th} cubic Bézier curve. A Similar expression follows for a linear or quadratic curve. The superscript denotes the frame where the variable is defined. Throughout this paper, it should be remembered that each curve has an associated polynomial order, which can easily be determined using a lookup table associated with the variable j.
$\mathbf{t}$	We define $\mathbf{t}$ as an ordered vector with n elements that are evenly spaced between zero and one, i.e., $\mathbf{t} = [0, \Delta t, 2\Delta t, \ldots, 1]^\top$.
$\mathbf{U}_{i,\mathcal{O}_j}$	Let t_i be the i^{th} element of $\mathbf{t}$, then $\mathbf{U}_{i,3} = \begin{bmatrix} t_i^3 & t_i^2 & t_i & 1 \end{bmatrix}^\top$, where the subscript denotes the order of the j^{th} curve . A similar expression follows for a linear or quadratic Bézier curve.
$\mathcal{A}(\mathbf{t}, \mathbf{C}_j^{\mathcal{W}})$	The linear transformation that maps $\mathbf{t}$ to points that lie on the bezier curve $\mathbf{C}_j^{\mathcal{W}}$, $\mathcal{A}(\mathbf{t}, \mathbf{C}_j^{\mathcal{W}})$ is defined in Section 2.3.
$\boldsymbol{\beta}$	$\boldsymbol{\beta}$ represents an estimated parameter vector of m Bézier curves, i.e., $\boldsymbol{\beta} \triangleq [(\mathbf{C}_1^{\mathcal{B}})^\top \ldots (\mathbf{C}_m^{\mathcal{B}})^\top]^\top$.

$$\mathcal{A}(t_i, \mathbf{C}_j^{\mathcal{W}}) = [\ \mathbf{P}_{j,0}^{\mathcal{W}},\ \mathbf{P}_{j,1}^{\mathcal{W}},\ \mathbf{P}_{j,2}^{\mathcal{W}},\ \mathbf{P}_{j,3}^{\mathcal{W}}\ ]\ \mathbf{B}_{\mathcal{O}_j}\mathbf{U}_{i,\mathcal{O}_j} \tag{2}$$

where $\mathbf{U}_{i,\mathcal{O}_j} = \begin{bmatrix} t_i^{\mathcal{O}_j} & t_i^{\mathcal{O}_j-1} & \dots & 1 \end{bmatrix}^\top$, and the matrix of constant coefficients $\mathbf{B}_{\mathcal{O}_j}$ is obtained from the Bernstein polynomial coefficients (see [Piegl and Tiller, 1997]). In fact, (2) is true for any $t \in [0, 1]$, but in this article, we are only concerned with mapping the uniformly spaced vector $\mathbf{t}$ onto a Bézier curve.

- Bézier curves are invariant under affine transformations, i.e., any affine transformation on a Bézier curve is equivalent to an affine transformation on the control points [Salomon, 2006].
- If a Bézier curve cannot be degree reduced, the control points are unique and the weights can only be varied in a known way by a perspective transformation [Berry and Patterson, 1997] [Patterson, 1985].

3 Estimating 3-D Body Frame Curves with a Single Stereo Pair

The purpose of this section is to demonstrate how to reconstruct the 3-D location of the path boundary using a single stereo pair. This task is accomplished by formulating and solving a nonlinear least-squares optimization problem that minimizes the reprojection error of the image coordinates comprising the path boundary. The optimization problem depends on two inputs, and the purpose of this section is to explain how to construct these two inputs. The first input is represented by the variable $\mathbf{y}^{o,j,i}$, which represents 2-D image coordinates located on the path boundary, where $o \in \{L, R\}$, j represents the j^{th} curve, and i represents the i^{th} discretized point belonging to curve j. The second input is the predicted measurement function $\hat{\mathbf{y}}^{o,j,i}(\cdot)$ that represents the projection of a 3-D curve from the body frame to the image plane. A high-level overview of the steps taken to construct these two inputs is as follows (these steps and two inputs are illustrated in Figure 5):

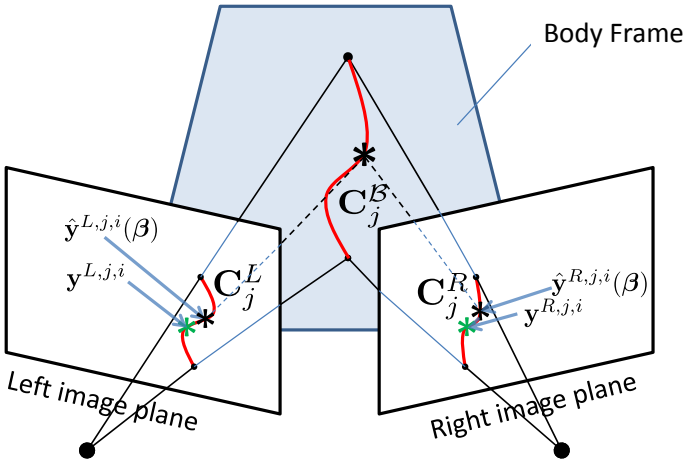
1. Locate the path boundary in each image of the stereo pair.
2. Interpolate and match m bezier curves $\mathbf{C}_1^L \dots \mathbf{C}_m^L$, $\mathbf{C}_1^R \dots \mathbf{C}_m^R$ to the path boundary in the stereo images; a single point located on these interpolated and matched curves represents the measurement $\mathbf{y}^{o,j,i}$.
3. Obtain the predicted measurement function $\hat{\mathbf{y}}^{o,j,i}(\cdot)$ by projecting points located on 3-D body frame curves $\mathbf{C}_1^{\mathcal{B}} \dots \mathbf{C}_m^{\mathcal{B}}$ to the image plane. The curve $\mathbf{C}_j^{\mathcal{B}}$ is related to the j^{th} curve in the world frame $\mathbf{C}_j^{\mathcal{W}}$, an element of the SLAM state defined in equation 1, as follows: $\mathbf{C}_j^{\mathcal{B}} = \mathbf{T}_{\mathcal{BW}}\mathbf{C}_j^{\mathcal{W}}$, where $\mathbf{T}_{\mathcal{BW}} \in SE(3)$. Additionally The curves $\mathbf{C}_1^L \dots \mathbf{C}_m^L$, $\mathbf{C}_1^R \dots \mathbf{C}_m^R$ correspond to the projection of $\mathbf{C}_1^{\mathcal{B}} \dots \mathbf{C}_m^{\mathcal{B}}$ in the stereo pair. This projection is uniquely defined. Furthermore, with the correct correspondence between the curves in $\mathbf{C}_1^L \dots \mathbf{C}_m^L$ and $\mathbf{C}_1^R \dots \mathbf{C}_m^R$, we are able to estimate the 3-D location of $\mathbf{C}_1^{\mathcal{B}} \dots \mathbf{C}_m^{\mathcal{B}}$ in the body-frame. A proof of these facts is presented in the appendix.

3.1 Extraction of Path Boundary

In this paper, we employ two methods to extract the boundary of the path. In the first method, we filter the image with an averaging filter to remove noise, threshold the image to locate the path, and apply a contour detector [Suzuki and Abe, 1985] that finds and sorts the path boundary according to spatial proximity. We repeat this procedure for both the left and right image of each stereo frame. The size ρ_n of the average filter window and image threshold are discussed in Section 5.4. The second method relies on a pre-trained convolutional neural network to detect pixels that represent the road [Teichmann et al., 2016]. Once the

(a) Step 1. Extract the boundary of the path

(b) Step 2. Interpolate and Match curves



(c) Step 3. Reconstruct the 3-D location of control Points

Figure 5: The steps taken to reconstruct the 3-D location of control points with respect to the body-frame. Also, see Proposition 1 in the Appendix

road is detected, we apply a contour detector [Suzuki and Abe, 1985] that finds and sorts the road boundary according to spatial proximity. We repeat this procedure for both the left and right images of each stereo frame.

3.2 Interpolating and Matching Curves in the Image Plane

To find $\mathbf{y}^{o,j,i}$, we interpolate and match a set of m Bézier curves $\mathbf{C}_1^L \dots \mathbf{C}_m^L$, $\mathbf{C}_1^R \dots \mathbf{C}_m^R$ to the path boundary in the stereo pair using linear least squares by modifying the algorithm in [Khan, 2007]. During this process, we must be able to quickly match a measurement $\mathbf{y}^{o,j,i}$ to a predicted measurement $\hat{\mathbf{y}}^{o,j,i}(\cdot)$, and determine where to split curves when the boundary is not sufficiently smooth. In this subsection, we explain how to construct $\mathbf{y}^{o,j,i}$ to accomplish these tasks. We define B as the set of image coordinates comprising the path boundary, and a break point as a single image coordinate belonging to the set B. A break point designates a desired start or end control point of the j^{th} curve $\mathbf{C}_j^o$ in the image plane. Between two break points, we

attempt to interpolate $\mathbf{C}_j^o$. Break points are determined when a curve is added to the state (Section 4.2), or from the data association step (Section 4.1). The steps to interpolate and match curves are as follows:

Step 1 Let $(u^r_{b_{2j-1}}, v^r_{b_{2j-1}})$ and $(u^r_{b_{2j}}, v^r_{b_{2j}})$ be the break points of curve j in image frame r and $B_c = \{(u^r_{b_{2j-1}}, v^r_{b_{2j-1}}), \ldots, (u^r_{b_{2j}}, v^r_{b_{2j}})\} \subset B$ be the image coordinates between the break points $(u^r_{b_{2j-1}}, v^r_{b_{2j-1}})$ and $(u^r_{b_{2j}}, v^r_{b_{2j}})$. We interpolate a single Bézier curve $\mathbf{C}_j^L$ to B_c by fixing the start control point $\mathbf{P}^L_{j,0}$ and end control point $\mathbf{P}^L_{j,3}$ at the break points. In other words, $\mathbf{P}^L_{j,0} = (u^r_{b_{2j-1}}, v^r_{b_{2j-1}})$, $\mathbf{P}^L_{j,3} = (u^r_{b_{2j}}, v^r_{b_{2j}})$, and depending on the order of the curve (determined in Section 4.3), we find the middle control points $\mathbf{P}^L_{j,1}$ and $\mathbf{P}^L_{j,2}$ with least-squares. We repeat this process for all the break points in the left image.

Step 2 Match curves between the left and right images. To do so, we obtain $\mathcal{A}(\mathbf{t}, \mathbf{C}_j^L)$, which has the effect of discretizing $\mathbf{C}_j^L$ into a set of spatially ordered points that lie on curve $\mathbf{C}_j^L$. In general, the spacing of these discretized points is not uniform, but is obtained by mapping the uniformly spaced vector $\mathbf{t}$ onto a nonlinear polynomial (see (2)). Other parameterizations exist for the vector $\mathbf{t}$ that allow the discretized curve points to be more uniformly spaced [Wang et al., 2002, Walter and Fournier, 1996], but these approaches are more computationally complex. Each point $\boldsymbol{\mathcal{P}}^L \in \mathcal{A}(\mathbf{t}, \mathbf{C}_j^L)$ must satisfy the epipolar constraint in the right image and must lie on the path boundary. These two constraints combined give a good initial estimate of where the discretized curve $\mathcal{A}(\mathbf{t}, \mathbf{C}_j^L)$ is located in the right image. Then, for each point $\boldsymbol{\mathcal{P}}^L \in \mathcal{A}(\mathbf{t}, \mathbf{C}_j^L)$, we implement a template matcher to further refine the curves location in the right image. The template matcher compares two small image patches between the left and right image. One image patch is centered around each point $\boldsymbol{\mathcal{P}}^L \in \mathcal{A}(\mathbf{t}, \mathbf{C}_j^L)$, while the second, slightly larger image patch is centered around the estimated location of $\boldsymbol{\mathcal{P}}^L$ in the right image. The size ρ_t of the template window is described in Section 5.4.

Step 3 Our last step is to interpolate a Bézier curve $\mathbf{C}_j^R$ to the refined location of $\mathcal{A}(\mathbf{t}, \mathbf{C}_j^L)$ in the right image.

Typical results of the matching process are shown in Figure 6. With the interpolated image curves, we find a measurement $\mathbf{y}^{o,j,i}$ by mapping the vector $\mathbf{t}$ with curve $\mathbf{C}_j^o$ to the image plane. A measurement is given by

$$\mathbf{y}^{o,j,i} = \mathcal{A}(t_i, \mathbf{C}_j^o) \tag{3}$$

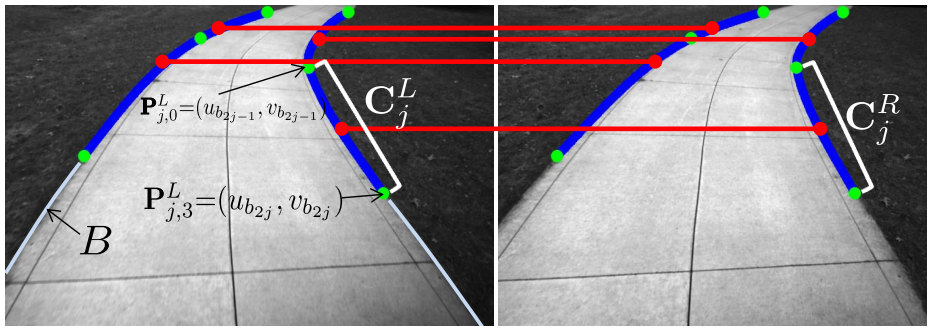


Figure 6: Curves matched in the left and right images. The green points represent the start or end point of a single curve. The red lines denote matching sets of curves.

3.3 Curve Parameter Optimization

Our next step is to reconstruct the 3-D coordinates of the path boundary with a series of m Bézier curves $\mathbf{C}_1^{\mathcal{B}} \dots \mathbf{C}_m^{\mathcal{B}}$. When reconstructing the path boundary, it is helpful to remember that the measured curves $\mathbf{C}_1^L \dots \mathbf{C}_m^L$, $\mathbf{C}_1^R \dots \mathbf{C}_m^R$, defined in Section 3.2, correspond to the projection of $\mathbf{C}_1^{\mathcal{B}} \dots \mathbf{C}_m^{\mathcal{B}}$ in the stereo pair. Defining $\boldsymbol{\beta} \triangleq [(\mathbf{C}_1^{\mathcal{B}})^\top \dots (\mathbf{C}_m^{\mathcal{B}})^\top]^\top$ as a stacked parameter vector of curve control points, we estimate $\boldsymbol{\beta}$ by formulating a nonlinear least-squares optimization problem that minimizes the reprojection error of the image coordinates comprising the path boundary in a single stereo frame. The general form of the optimization problem is given by

$$\boldsymbol{\beta} = \arg\min_{\boldsymbol{\beta}} \sum_{o \in \{L,R\}} \sum_{j=1}^{m} \sum_{i=1}^{n} ||\mathbf{y}^{o,j,i} - \hat{\mathbf{y}}^{o,j,i}(\boldsymbol{\beta})||^2 \tag{4}$$

where o indicates the left or right stereo image, j indicates the curve, and i indicates the image coordinates belonging to curve j. The 2-D vector $\mathbf{y}^{o,j,i}$ represents measured image coordinates belonging to the path boundary. We showed how to calculate $\mathbf{y}^{o,j,i}$ in Section 3.2. The function $\hat{\mathbf{y}}^{o,j,i}(\cdot)$ represents a measurement predicted by the parameters in $\boldsymbol{\beta}$. To find a predicted measurement, we map the vector $\mathbf{t}$, using the j^{th} curve $\mathbf{C}_j^{\mathcal{B}}$, from the body frame to the image plane. Letting t_i be the i^{th} element of $\mathbf{t}$, a predicted measurement is given by the equation

$$\hat{\mathbf{y}}^{o,j,i}(\boldsymbol{\beta}) = \mathbf{M}^o(\mathcal{A}(t_i, \mathbf{C}_j^{\mathcal{B}})) \tag{5}$$

where $\mathbf{M}^o(\cdot)$ is the function that maps a 3-D point from the body frame to the image plane of the left or right camera.

With the measurement and predicted measurement defined, we solve (4) with the Levenberg-Marquardt algorithm [Levenberg, 1944] [Marquardt, 1963] [Agarwal et al., 2015]. An initial guess for the parameters in $\boldsymbol{\beta}$ is given by the optimized variables from the previous image frame. Once the optimization is complete, the reprojection error of each curve is checked. Any curve with a reprojection larger than a threshold ρ_r is discarded. The selection of ρ_r is based on the precision of the stereo camera calibration. The purpose of ρ_r is simply to provide a fail safe when it is obvious the stereo triangulation failed, the size of ρ_r is discussed in Section 5.4. The output of the Levenberg-Marquardt optimization gives an estimate of the curve control points defined with respect to the body frame.

After applying the Levenberg-Marquardt algorithm to estimate $\boldsymbol{\beta}$, we calculate the measurement covariance matrix $\mathbf{V}_r$, as described in Section 5.4 [Myers et al., 2010]. The covariance $\mathbf{V}_r$ is used as the extended Kalman Filter measurement covariance in Section 4.6, and will be used in solving the data association step in Section 4.1.

4 SLAM Estimator Formulation

In this section, we propose a solution to the data association problem and formulate an Extended Kalman filter to solve the SLAM problem. To do so, we determine the order of each Bézier curve and determine when to add curves to the filter state. This section explains how to accomplish these tasks.

4.1 Curve-Based Data Association

We solve the data association problem by tracking the start and end points of curves between sequential frames in the left camera's field of view (FOV), and by comparing the 3-D structure of these curves between

frames to ensure they were tracked correctly. Tracking is done with the the Lucas-Kanade tracking (KLT) algorithm [Lucas and Kanade, 1981]. In implementing the KLT algorithm, it is important to emphasize that our data association algorithm is different from conventional point-based tracking algorithms, e.g., the type that relies on salient regions in the image to detect, describe, and track a uniform distribution of points between image frames, followed by an outlier rejection algorithm to remove outliers, see [Luetenegger et al., 2015] as an example. Instead, the KLT algorithm is used to track just the start or end points of curves between two sequential image frames. Additionally, because we remove outliers by comparing the 3-D shape of curves between image frames, it is sufficient to track fewer than five landmark curves between most image frames. In other words, our algorithm is not dependent on tracking a large number of point-based landmarks. In turn, this allows our algorithm to operate in settings that lack distinguishable feature points and to create maps that are more sparse than point-based SLAM algorithms. This approach is quite different from point-based data association algorithms that track hundreds of feature points between image frames. Finally, our data association step is different from tracking a collection of feature points because curves lie on the edge of a path. Thus, when finding a curve correspondence between two chronologically sequential image frames, the data association search space is limited to a one-dimensional edge.

The KLT algorithm will occasionally track the start and end control points to the wrong location, producing outliers. We remove curve outliers by comparing their 3-D curve shapes between frames. We explain our outlier rejection process assuming cubic ordered curves since similar steps can be applied to linear or quadratic curves. To compare curves, we define $\mathbf{d}_{l,l+1}^{r-1}$ as the estimated distance between two control points, i.e., $\mathbf{d}_{l,l+1}^{r-1} = \mathbf{P}_{j,l}^{\mathcal{B}_{r-1}} - \mathbf{P}_{j,l+1}^{\mathcal{B}_{r-1}}$, see Figure 7. Alternatively, $\mathbf{d}_{l,l+1}^{r-1}$ can be written as $\mathbf{d}_{l,l+1}^{r-1} = \boldsymbol{A}\boldsymbol{\beta}$, where $\boldsymbol{A}$ is of the form $\boldsymbol{A} = [0, \ldots, 0, 1, -1, 0, \ldots, 0]$. In this case, $\boldsymbol{\Sigma}_{l,l+1}^{r-1}$ has an estimated covariance given by $\boldsymbol{\Sigma}_{l,l+1}^{r-1} = \boldsymbol{A}\mathbf{V}_{r-1}\boldsymbol{A}^{\top}$. The steps to solve the data association problem are as follows:

Step 1 Track the break points of curves $(u_{b_1}^{r-1}, v_{b_1}^{r-1}), \ldots, (u_{b_{2m}}^{r-1}, v_{b_{2m}}^{r-1})$ from frame $r-1$ to frame r with the the Lucas-Kanade tracking (KLT) algorithm. We represent the image coordinates that tracked from frame $r-1$ to frame r as $(u_{t_1}^{r}, v_{t_1}^{r}), \ldots, (u_{t_{2m}}^{r}, v_{t_{2m}}^{r})$. These image coordinates are not yet confirmed as break points because the KLT algorithm may fail to correctly track break points between image frames.

Step 2 Track the image coordinates $(u_{t_1}^{r}, v_{t_1}^{r}), \ldots, (u_{t_{2m}}^{r}, v_{t_{2m}}^{r})$ from frame r back to frame $r-1$, forming the image coordinates $(u_{t_1}^{r-1}, v_{t_1}^{r-1}), \ldots, (u_{t_{2m}}^{r-1}, v_{t_{2m}}^{r-1})$.

Step 3 For each break point in frame $r-1$, measure the Euclidean distance between the location of the break point and the location the break point is tracked to in step 2, i.e., for the q^{th} breakpoint determine $||(u_{t_q}^{r-1}, v_{t_q}^{r-1}) - (u_{b_q}^{r-1}, v_{b_q}^{r-1})||$. If a curve's break point is not tracked to its original location then the curve is removed.

Step 4 The remaining tracked points are assigned as break points and the curve fitting algorithm of Section 3 is computed for frame r, outputting the 3-D location of curves with respect to the r^{th} body frame.

Step 5 Verify that the 3-D curve structure matches between frame r and frame $r-1$ (see Figure 7). We verify this by performing two Mahalanobis distance tests. The first Mahalanobis distance test verifies the distance between all sequential control points with the estimated covariance of $\mathbf{d}_{l,l+1}^{r-1}$ and sample $\mathbf{d}_{l,l+1}^{r}$. The second Mahalanobis distance test verifies the distance between the start and end point of the curve using the estimated covariance of $\mathbf{d}_{0,3}^{r-1}$ and sample $\mathbf{d}_{0,3}^{r}$. Note, in the second Mahalonobis distance test, we include a max threshold tolerance ρ_s. If a curve changes shape by more than ρ_s, it fails the second Mahalonobis distance test. The selection of ρ_s is based on the precision of the stereo camera. The purpose of ρ_s is simply to provide a fail safe when it is obvious the KLT algorithm tracked feature points incorrectly. The size of ρ_s is discussed in Section 5.4.

Step 6 If the curve fails the second Mahalonobis distance test in Step 5, the curve is removed from the state. If the curve passes the second Mahalanobis distance test but not the first, the curve is treated as a linear Bézier curve.

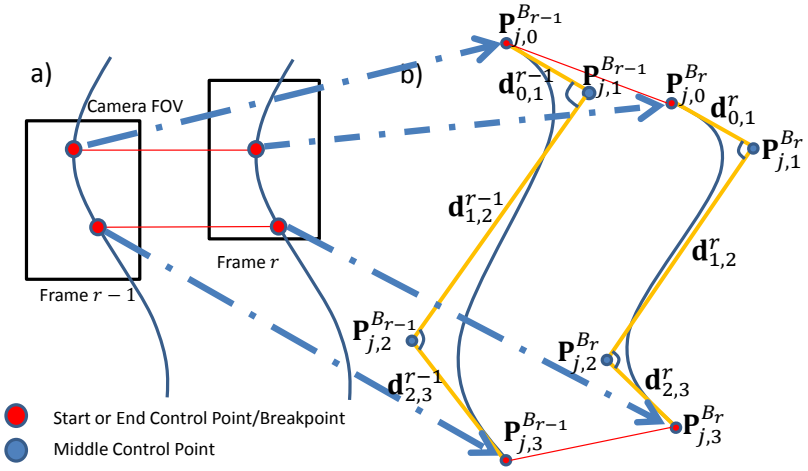


Figure 7: a) Tracking endpoints of a curve between two chronologically sequential frames. b) comparing the structure of matched curves, this particular example demonstrates a tracking failure.

4.2 Adding Curves in the Image Plane

Curves are added so that their lengths are approximately equivalent. We add a curve when the end control point of the currently tracked curve is about to leave the camera's FOV. This illustrated in Figure 8a by the tracked curve crossing the the yellow line L_e. When this occurs, a desired control point is set approximately at the top of the camera's FOV and at the start control point of the currently tracked curve (see Figure 8a). In the initial frame or when a particular side of the boundary is empty of curves (a particular side may be empty of curves due to tracking failures), we arbitrarily set a desired control point approximately at the top of the camera's FOV, the bottom of the camera's FOV, and at half the arc length of the path boundary. These desired control points represent a start or end point of a curve. A region of interest is selected around the desired control points, and the Shi-Tomasi corner detector [Shi and Tomasi, 1994] is used to find good features to track in this region. Corner points whose distance exceeds a threshold ρ_k from the path boundary are rejected. Among the remaining corner points in a single region, the point with the strongest corner feature is selected as the new start or end control point (see Figure 8b). The size of the region of interest ρ_w and the parameter ρ_k are discussed in Section 5.4.

4.3 Automatic Correction of the Polynomial Curve Order

With the newly determined start or end points (determined in the previous subsection, see Section 4.2), we are ready to determine the polynomial order of these newly added curves. Determining the polynomial order is necessary for two reasons: First, when adding curves to the boundary of the path, care must be taken to avoid over-fitting the path by interpolating a higher order curve to the boundary when only a lower order curve is required. Otherwise, it is likely that the fitted 3-D control points will not remain static between image frames, causing localization errors. Second, we need a way to determine when to split these newly added curves when the boundary of the path is not sufficiently smooth (see Figure 9). This section explains how to accomplish both of these tasks. To determine the order of a curve, let $\mathbf{P}_{j,0}^{L}$ $\mathbf{P}_{j,3}^{L}$ be the start and end points of a newly added curve found in Section 4.2. Additionally, let $B_c = \{\mathbf{P}_{j,0}^{L}, \ldots, \mathbf{P}_{j,3}^{L}\} \subset B$ represent the image coordinates of the path boundary between $\mathbf{P}_{j,0}^{L}$ and $\mathbf{P}_{j,3}^{L}$. Starting with a first order curve $\mathcal{O}_j = 1$, the following steps are taken to determine the curve's order:

Step 1 Given $\mathcal{O}_j$, interpolate a Bézier curve of order $\mathcal{O}_j$ to the boundary of the path between the newly added control points $\mathbf{P}_{j,0}^{L}$ $\mathbf{P}_{j,3}^{L}$.

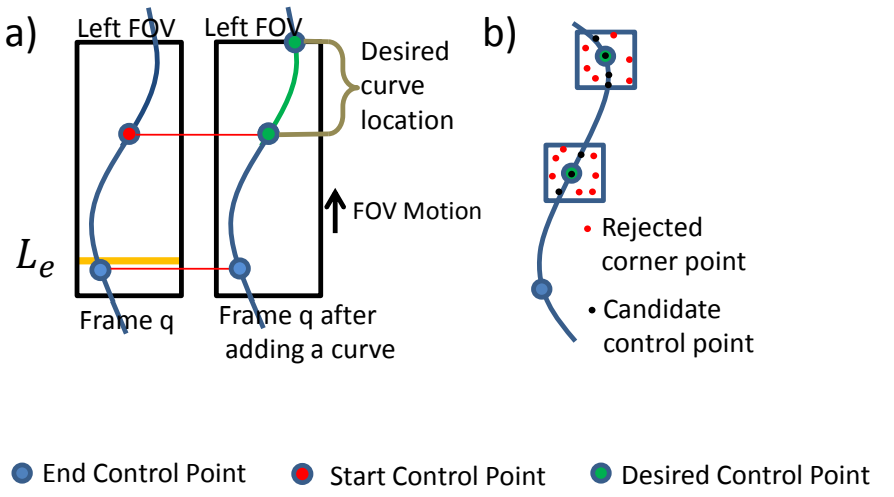


Figure 8: Events triggering the addition of a curve to the state and their resulting addition. A curve is added when the the currently tracked curve is about to leave the camera's FOV, see Figure 8a. When this occurs, a region of interest is set around the location of desired control points, and the Shi-Tomasi corner detector [Shi and Tomasi, 1994] is used to find good features to track in these regions see Figure 8b (the Shi-Tomasi corner detector allows the KLT tracking algorithm to track start/end control points with greater accuracy). The point with the strongest corner feature in this region that is close enough to the boundary of the path is selected as the new start or end control point.

Step 2 Using the Shapiro-Wilk test [Shapiro and Wilk, 1965], check that the residuals of the interpolated curve from Step 1 are normally distributed. If the residuals are normally distributed, the curve order has been determined. If the residuals are not normally distributed, increase the curve order by one, i.e., $\mathcal{O}_j = \mathcal{O}_j + 1$. Note, in this step, we also include a minimum threshold tolerance ρ_p to avoid over-splitting the path boundary (splitting is discussed in Step 4 of this Section). As long as the maximum residual of the interpolated curve from Step 1 is below ρ_p, the order of the curve has been determined. We discuss the parameter ρ_p in Section 5.4.

Step 3 Repeat Step 1–Step 2, progressing from a first-order curve to a third-order curve. If the curve is not third-order, proceed to Step 4.

Step 4 If the curve is not third-order, we split B_c at the point on B_c where the residual is a maximum, and designate this split point (u_s, v_s) as a new break point. This forms two sets of data points $B_1 = \{\mathbf{P}^L_{j,0}, \ldots, (u^r_s, v^r_s)\}$ and $B_2 = \{(u^r_s, v^r_s), \ldots, \mathbf{P}^L_{j,3}\}$, where B_1 consists of all the ordered points before the break point, and B_2 consists of all the ordered points after the break point.

Step 5 Repeat Step 1–Step 3 recursively on the data points in B_1 and B_2 until the residuals are normally distributed.

The start and end points of curves determined from this process are used as as break points in Section 3.2.

4.4 SLAM Estimator State and Sensors

We estimate the camera pose, linear velocity, and location of curve control points with an extended Kalman filter (EKF). We define the variable $\boldsymbol{P}$ as the error covariance matrix, and the variable $\mathbf{x}$ as the filter state:

$$\mathbf{x} \triangleq [\mathbf{p}^{\mathcal{W}}, \mathbf{v}^{\mathcal{B}}, \boldsymbol{\Theta}, \mathbf{b}_a, \mathbf{b}_g, (\mathbf{C}^{\mathcal{W}}_1)^\top, \ldots, (\mathbf{C}^{\mathcal{W}}_N)^\top]^\top$$

(a) A path with sharp corners.

(b) Curves automatically split and interpolated to the path.

Figure 9: Figure 9a demonstrates a path that contains sharp corners and is not sufficiently smooth for a single cubic Bézier curve. When this occurs, the Curve SLAM algorithm will automatically split the boundary of the path, see Figure 9b.

where the variables $\mathbf{p}^{\mathcal{W}}$, $\mathbf{v}^{\mathcal{B}}$, and $\boldsymbol{\Theta}$ are defined in Table 1. The variables $\mathbf{b}_a \in \mathbb{R}^3$ and $\mathbf{b}_g \in \mathbb{R}^3$ represent the acclerometer bias and gyroscope bias, respectively.

An image of the sensors used to collect experimental data for Curve SLAM is shown in Figure 10. In addition to a stereo camera, our hardware is equipped with a Novatel SPAN-IGM-A1 GNSS inertial navigation system equipped with a GPS and an Analog Devices ADIS 16488 IMU consisting of an accelerometer and gyroscope. The GPS is used only for ground truth. All hardware has been calibrated so that measurements can be transformed to the body frame of the left camera [Furgale et al., 2013, Maye et al., 2013, Furgale et al., 2012]. The accelerometer and gyroscope are used to propagate the state equations in the prediction step of the EKF, where the gyroscope bias and accelerometer bias are both propagated as a random walk. The gyroscope measurement $\boldsymbol{\omega}_k^m$ at time step k is given by $\boldsymbol{\omega}_k^m = \boldsymbol{\omega}_k + \boldsymbol{\omega}_k^n + \mathbf{b}_g$, where $\boldsymbol{\omega}_k$ is the true angular rate of the camera, and $\boldsymbol{\omega}_k^n$ represents gyroscope noise. The accelerometer measurement $\mathbf{a}_k^m$ at time step k is given by $\mathbf{a}_k^m = \dot{\mathbf{v}}^{\mathcal{B}} + \boldsymbol{\omega}_k \times \mathbf{v}^{\mathcal{B}} - \boldsymbol{R}_{\mathcal{BW}}(\boldsymbol{\Theta})\mathbf{g}^{\mathcal{W}} + \mathbf{a}_k^n + \mathbf{b}_a$, where $\mathbf{a}_k^n$ represents accelerometer noise. Letting $\mathbf{a}_k = \dot{\mathbf{v}}^{\mathcal{B}} + \boldsymbol{\omega}_k \times \mathbf{v}^{\mathcal{B}} - \boldsymbol{R}_{\mathcal{BW}}(\boldsymbol{\Theta})\mathbf{g}^{\mathcal{W}}$, we define $\mathbf{u}_k = [\mathbf{a}_k^\top \ \boldsymbol{\omega}_k^\top]^\top$ as the input vector, and $\mathbf{w}_k = [\mathbf{a}_k^{n\top} \ \boldsymbol{\omega}_k^{n\top} \ \mathbf{b}_k^{n,a} \ \mathbf{b}_k^{n,g}]^\top$ as the process noise with covariance matrix $\mathbf{W}$. The variables $\mathbf{b}_k^{n,a} \in \mathbb{R}^3$, $\mathbf{b}_k^{n,g} \in \mathbb{R}^3$ represent accelerometer and gyroscope bias noise, respectively. This noise comes as a result of propagating the bias states as a random walk.

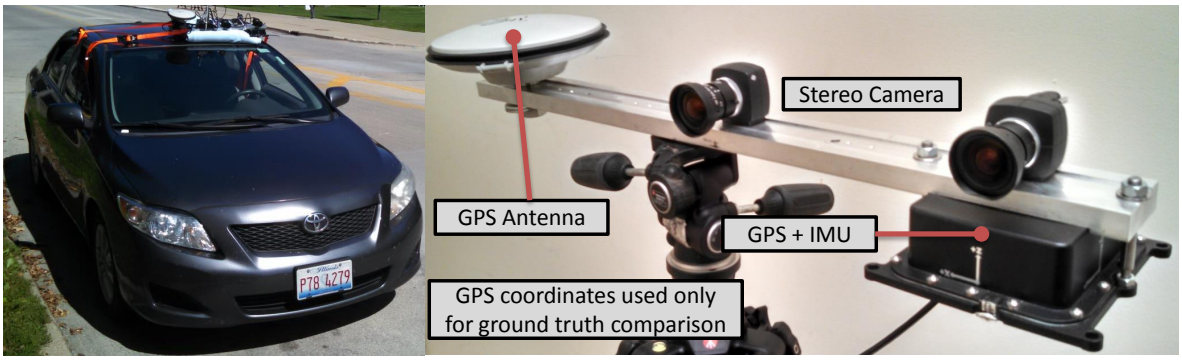


Figure 10: Our sensor package consists of a stereo camera, an IMU and GPS

4.5 Estimator Prediction Step

We implement the EKF prediction step by feeding the data from the IMU as a dynamic model replacement, where the gyroscope bias and accelerometer bias are both propagated as a random walk [Trawny and

Roumeliotis, 2005]. To explain the process, we adopt standard EKF notation in which the subscript $k|k-1$ represents a predication step, while the subscript $k|k$ represents a measurement update. Using one-step Euler integration, the predicted state $\hat{\mathbf{x}}$ at time step k is given by $\hat{\mathbf{x}}_{k|k-1} = \hat{\mathbf{x}}_{k-1|k-1} + f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}, \mathbf{w}_{k-1})\Delta t$ where

$$f(\mathbf{x}, \mathbf{u}_k, \mathbf{w}_k) = \begin{pmatrix} \boldsymbol{R}_{\mathcal{WB}}(\boldsymbol{\Theta})\mathbf{v}^{\mathcal{B}} \\ \mathbf{a}_k^m - \mathbf{b}_a - (\boldsymbol{\omega}_k^m - \mathbf{b}_g) \times \mathbf{v}^{\mathcal{B}} + \boldsymbol{R}_{\mathcal{BW}}(\boldsymbol{\Theta})\mathbf{g}^{\mathcal{W}} \\ \boldsymbol{S}(\boldsymbol{\Theta})(\boldsymbol{\omega}_k^m - \mathbf{b}_g) \\ \mathbf{0}^{3\times 1} \\ \mathbf{0}^{3\times 1} \\ \mathbf{0}^{N\times 1} \end{pmatrix} \tag{6}$$

The variable $\boldsymbol{R}_{\mathcal{WB}}(\Theta)$ is a rotation matrix from the body frame to the world frame, and $\boldsymbol{S}(\boldsymbol{\Theta})$ is a rotational transformation that allows the body frame angular rates to be expressed in terms of the derivatives of the ZYX Euler angles, i.e.,

$$\boldsymbol{S}(\boldsymbol{\Theta}) = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{pmatrix} \tag{7}$$

The error covariance is updated as

$$\boldsymbol{P}_{k|k-1} = \mathbf{F}_{k-1}\boldsymbol{P}_{k-1|k-1}\mathbf{F}_{k-1} + \mathbf{L}_{k-1}\mathbf{W}\mathbf{L}_{k-1}^{\top}$$

where $\mathbf{F}_k = \frac{\partial f(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k, \mathbf{w}_k)}{\partial \mathbf{x}}$ and $\mathbf{L}_k = \frac{\partial f(\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k, \mathbf{w}_k)}{\partial \mathbf{w}}$. We compute the expressions for the Jacobians $\mathbf{F}_k$ and $\mathbf{L}_k$ symbolically offline.

4.6 Measurement Update

At every stereo frame, we measure m different Bézier curves $\mathbf{C}_1^{\mathcal{B}} \dots \mathbf{C}_m^{\mathcal{B}}$. These curves are related to the existing map curves by the transformation $\mathbf{C}_j^{\mathcal{B}} = \mathbf{T}_{\mathcal{BW}}(\mathbf{C}_j^{\mathcal{W}})$, where $\mathbf{T}_{\mathcal{BW}} \in SE(3)$ is the transformation that changes the representation of a point defined in the coordinates of frame $\mathcal{W}$ to a point defined in the coordinates of frame $\mathcal{B}$. The measurement equation is given by $\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k$, where

$$\mathbf{h}(\mathbf{x}) = [(\mathbf{T}_{\mathcal{BW}}(\mathbf{C}_1^{\mathcal{W}}))^{\top}, \dots, (\mathbf{T}_{\mathcal{BW}}(\mathbf{C}_m^{\mathcal{W}}))^{\top}]^{\top}. \tag{8}$$

The variable $\mathbf{v}_k$ represents measurement noise with covariance matrix $\mathbf{V}_k$. The measurement covariance matrix $\mathbf{V}_k$ is defined as stated in Section 3.2. We demonstrate how to calculate $\mathbf{V}_k$ in Section 5.4. The remaining EKF update equations are implemented as follows:

$$\begin{aligned} \tilde{\mathbf{y}}_k &= \mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) \\ \mathbf{S}_k &= \mathbf{H}_k\boldsymbol{P}_{k|k-1}\mathbf{H}_k^{\top} + \mathbf{V}_k \\ \mathbf{K}_k &= \boldsymbol{P}_{k|k-1}\mathbf{H}_k^{\top} + \mathbf{S}_k^{-1} \\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{y}}_k \\ \boldsymbol{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\boldsymbol{P}_{k|k-1} \end{aligned}$$

where $\mathbf{H}_k = \frac{\partial \mathbf{h}(\hat{\mathbf{x}}_{k|k-1})}{\partial \mathbf{x}}$. We compute the expression for the Jacobian $\mathbf{H}_k$ symbolically offline.

4.7 Adding Curve Control Points to the Filter State

After a curve has been added in the image plane (see Section 4.2), the filter state needs to be updated. With the method in [Pedraza et al., 2009], we augment the filter state with the new curve $\mathbf{C}_{N+1}^{\mathcal{B}}$ and augment

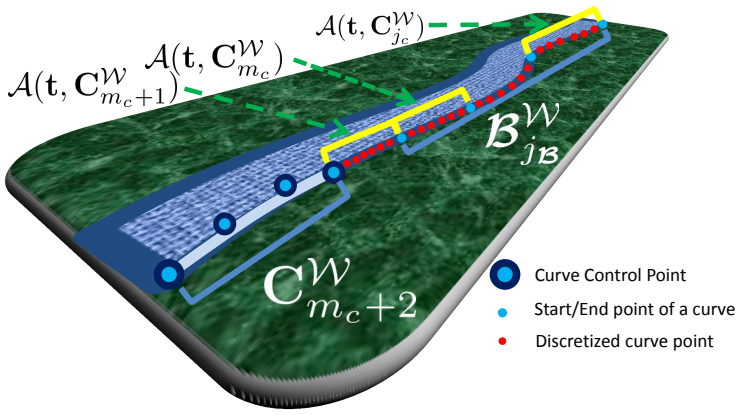


Figure 11: The variables used to describe the curve combining algorithm.

the error covariance matrix with the necessary initial cross-covariances. Letting $\mathbf{x}^a$ represent the augmented state and $\boldsymbol{P}^a$ represent the augmented error covariance, we perform this operation as follows:

$$\mathbf{x}^a = \mathbf{g}(\mathbf{x}, \mathbf{z})$$
$$\boldsymbol{P}^a = \mathbf{G}_{\mathbf{x}} \boldsymbol{P} \mathbf{G}_{\mathbf{x}}^\top + \mathbf{G}_{\mathbf{z}} \mathbf{V}_r \mathbf{G}_{\mathbf{z}}^\top$$

where $\mathbf{g}(\mathbf{x}, \mathbf{z}) = [\mathbf{x}^\top, \mathbf{T}_{\mathcal{W}\mathcal{B}}(\mathbf{C}^{\mathcal{B}}_{N+1})^\top]^\top$, $\mathbf{G}_{\mathbf{x}} = \frac{\partial \mathbf{g}(\mathbf{x},\mathbf{z})}{\partial \mathbf{x}}$, and $\mathbf{G}_{\mathbf{z}} = \frac{\partial \mathbf{g}(\mathbf{x},\mathbf{z})}{\partial \mathbf{z}}$. We compute the expressions for the Jacobians $\mathbf{G}_{\mathbf{z}}$ and $\mathbf{G}_{\mathbf{x}}$ symbolically offline.

4.8 Combining Curves

One of our main objectives is to represent long segments of a path with a small number of curves. However, because the depth measurement accuracy of a stereo camera is limited by range, our sparseness objective interferes with how accurately we are able to localize the camera. Thus, we limit the length of curve segments. To overcome this drawback, we add one additional step while mapping the environment. When the location of a curve $\mathbf{C}^{\mathcal{W}}_{m_c+1}$ is no longer contained in the camera's FOV, we attempt to combine $\mathbf{C}^{\mathcal{W}}_{m_c+1}$ with curves $\mathbf{C}^{\mathcal{W}}_{j_c} \dots \mathbf{C}^{\mathcal{W}}_{m_c}$ that were estimated prior to $\mathbf{C}^{\mathcal{W}}_{m_c+1}$. The steps to combine curves are given as follows:

Step 1 Assume $\mathbf{C}^{\mathcal{W}}_{j_c}, \dots, \mathbf{C}^{\mathcal{W}}_{m_c}$ belong to a single growing cubic Bézier curve $\boldsymbol{\mathcal{B}}^{\mathcal{W}}_{j_{\boldsymbol{\mathcal{B}}}}$ (see Figure 11).

Step 2 Interpolate a single cubic Bézier curve to $\mathcal{A}(\mathbf{t}, \mathbf{C}^{\mathcal{W}}_{j_c}) \dots \mathcal{A}(\mathbf{t}, \mathbf{C}^{\mathcal{W}}_{m_c})$ and $\mathcal{A}(\mathbf{t}, \mathbf{C}^{\mathcal{W}}_{m_c+1})$ by fixing the start control point at the start point in $\mathcal{A}(\mathbf{t}, \mathbf{C}^{\mathcal{W}}_{j_c})$ and the end control point at the last point in $\mathcal{A}(\mathbf{t}, \mathbf{C}^{\mathcal{W}}_{m_c+1})$. The two middle control point are determined with least squares.

Step 3a If the median of the residuals in the least squares fit is less than a threshold d, $\mathbf{C}^{\mathcal{W}}_{m_c+1}$ is added to $\boldsymbol{\mathcal{B}}^{\mathcal{W}}_{j_{\boldsymbol{\mathcal{B}}}}$. The size of d is discussed in Section 5.4

Step 3b If the median of the residuals in the least squares fit is not less than a threshold d, we start a new growing curve $\boldsymbol{\mathcal{B}}^{\mathcal{W}}_{j_{\boldsymbol{\mathcal{B}}}+1}$, with $\mathbf{C}^{\mathcal{W}}_{m_c+1}$ as its only element. Meanwhile, the past curve $\boldsymbol{\mathcal{B}}^{\mathcal{W}}_{j_{\boldsymbol{\mathcal{B}}}}$ is no longer growing, i.e., no curves that are subsequently estimated are added to $\boldsymbol{\mathcal{B}}^{\mathcal{W}}_{j_{\boldsymbol{\mathcal{B}}}}$. Instead, they are checked for addition to curve $\boldsymbol{\mathcal{B}}^{\mathcal{W}}_{j_{\boldsymbol{\mathcal{B}}}+1}$.

5 Experimental Results

We compare Curve SLAM against the Open Keyframe-based Visual Inertial SLAM algorithm (OKVIS) [Luetenegger et al., 2015], and a stereo version of the Parallel Tracking and Mapping (SPTAM) [Pire et al., 2015] [Klein and Murray, 2007] by adopting the metric proposed in [Geiger et al., 2012], and by comparing the number of landmarks required to represent the map. We used five different data sets that were captured at different of the time under varying environmental conditions. The first three data sets contain images of sidewalks that were obtained from a local park (the sidewalk provided curves for the Curve SLAM algorithm), the fourth data set contains images of yellow road lanes that were obtained while driving on a nearby road (the yellow road lanes provided curves for the Curve SLAM algorithm), and the fifth data set is sequence taken from the KITTI data set [Geiger et al., 2013], and contains images of a road (the road provided curves for the Curve SLAM algorithm).

Sample images of the five data sets are shown in Figures 12–16. During the first four data sets, the stereo camera had a 36 cm baseline with 3.5 mm focal length lens. Images were sampled at 20 Hz, and IMU measurements were sampled at 100 Hz. During the fifth data set, the stereo camera had a 54 cm baseline with 4 mm focal length lens. Images were sampled at 10 Hz, and IMU measurements were sampled at 10 Hz. During the first four data sets, we segmented the boundary of the path by thresholding the HSV color channels of an image (our approach is described in Section 3.1). During the fifth data set, we segmented the boundary of the path with a pre-trained convolutional neural network designed to detect image pixels that represent the road (see Section 3.1). The first three data sets were collected from a local park at different times of the day under different lighting and weather conditions. All the data in the park was obtained by mounting the sensors onto a cart that was pushed by a person. The edges of a long curved sidewalk in this local park were used as curves in the Curve SLAM algorithm. The first data set, DS1, was collected at dawn under clear weather conditions. The experiment lasted roughly 138.5 seconds. During this time, our sensors traveled approximately 252.2 meters. The second data set, DS2, was collected in the mid-afternoon, roughly 2 PM. Part of DS2 was collected during a light rainstorm under cloudy conditions, while the other part of DS2 was collected immediately following this light rainstorm under mostly sunny conditions. DS2 lasted roughly 195.85 seconds. During this time, our sensors traveled approximately 375 meters. DS3 was collected about an hour prior to sunset on a mostly sunny day with clear weather conditions. DS3 lasted roughly 437.55 seconds. During this time, our sensors traveled approximately 603.4 meters. The fourth data set DS4 was collected by strapping the sensors onto a car and driving on a nearby road. Yellow road lanes were used as curves. DS4 was collected in the morning hours under mostly cloudy conditions. DS4 lasted roughly 70 seconds. During this time, our sensors traveled approximately 230 meters. The fifth data set DS5 is a sequence obtained from the Kitti data set [Geiger et al., 2013], and was collected with the sensors attached to the top of a car while driving in a residential area. The edges of a road were used as curves. This particular sequence from the Kitti data set was selected due to the presence of curves in the environment and the high number of occlusions covering the road. DS5 was collected under sunny conditions. DS5 lasted roughly 40 seconds. During this time, the sensors traveled approximately 435 meters.

For a ground truth comparison of all the data sets, we have included a precise GPS-INS track that is time-synchronized with the stereo camera and IMU measurements. The GPS-INS track includes a ground-truth for the location and attitude of the sensor platform. Note, the GPS is only used to provide ground truth information.

5.1 Evaluation Metric

To compare Curve SLAM against OKVIS and SPTAM, we extend the metric proposed in [Geiger et al., 2012]. Letting d represent the distance traveled between frame i_e and frame j_e, we compare algorithms with the following metric:

$$\Delta\mathbf{T}(d) = \mathbf{T}_{0j_e}^{-1}\mathbf{T}_{0i_e}\hat{\mathbf{T}}_{0i_e}^{-1}\hat{\mathbf{T}}_{0j_e} \tag{9}$$

Figure 12: Sample images of DS1. The edges of the curved sidewalk provided curves for the Curve SLAM algorithm. Collection time: Dawn. Weather: mostly cloudy and clear. Length: 252.2 meters. Duration: 138.5 seconds.

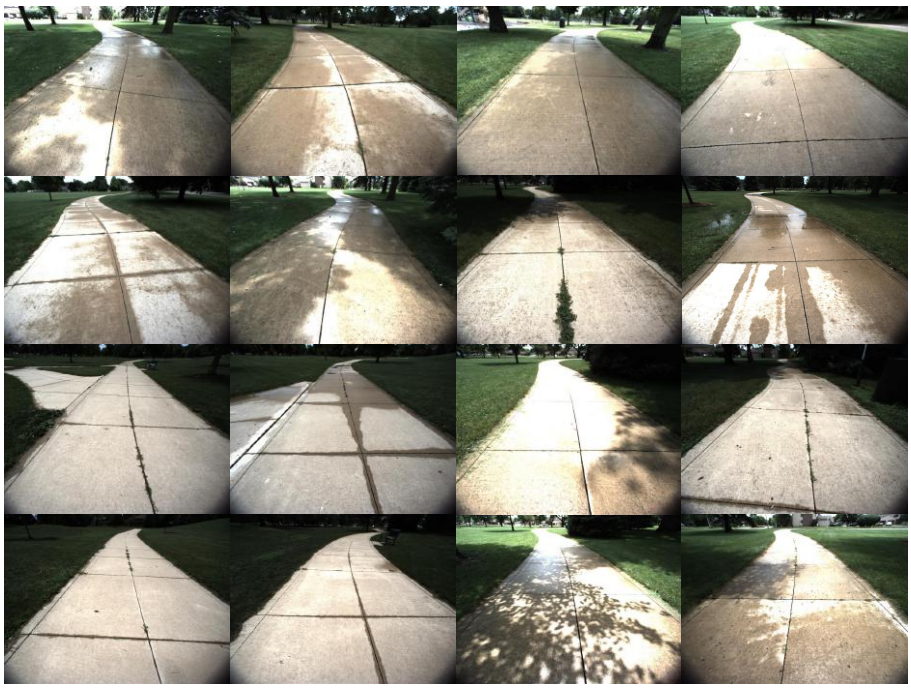


Figure 13: Sample images of DS2. The edges of the curved sidewalk provided curves for the Curve SLAM algorithm. Collection time: 2PM. Weather: Part of DS2 was collected during a light rainstorm under cloudy conditions. The other part was collected under mostly sunny conditions and clear weather. Length: 375 meters. Duration: 195.85 seconds.

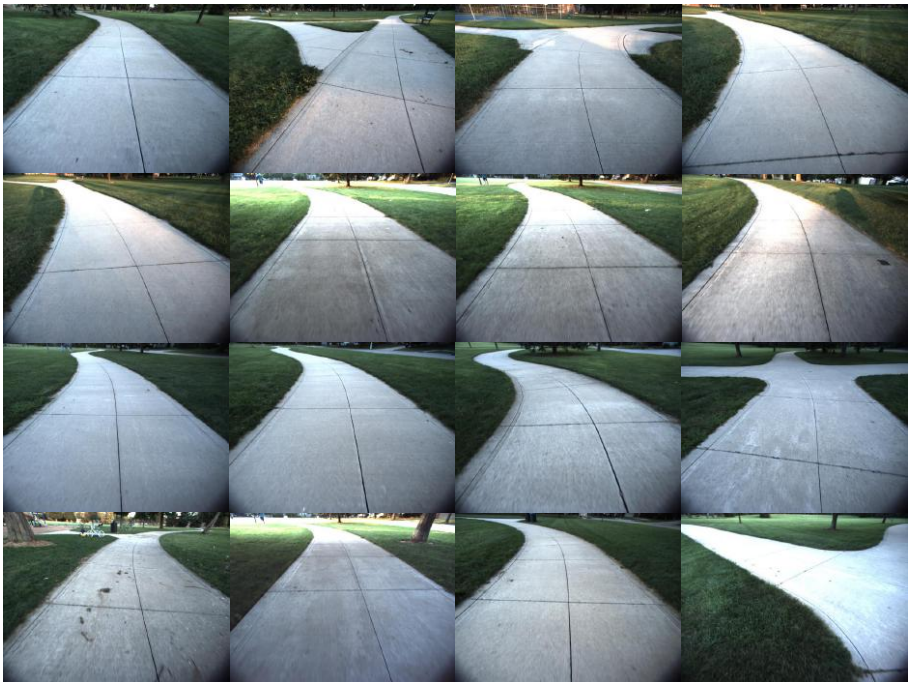


Figure 14: Sample images of DS3. The edges of the curved sidewalk provided curves for the Curve SLAM algorithm. Collection time: roughly one hour prior to sunset. Weather: mostly sunny and clear. Length: 603.4 meters. Duration: 437.55 seconds.

Figure 15: Sample images of DS4. Yellow road lanes provided curves for the Curve SLAM algorithm. Collection time: roughly 10:30 AM. Weather: mostly cloudy and clear. Length: 230 meters. Duration: 70 seconds.

Figure 16: Sample images of DS5. The edges of the road provided curves for the Curve SLAM algorithm. Collection time: unknown. Weather: sunny and clear. Length: 435 meters. Duration: 40 seconds.

where $\Delta\mathbf{T}(d) \in SE(3)$ represents the error between an estimated pose $\hat{\mathbf{T}}_{i_e j_e} \in SE(3)$ and ground truth pose $\mathbf{T}_{i_e j_e} \in SE(3)$, and the subscript 0 represents the initial frame. To compute d, we sum the euclidean distance between all temporally sequential GPS ground truth coordinates between frame i_e and frame j_e. For a fixed value of d, we compute $\Delta\mathbf{T}(d)$ between numerous poses i_e and j_e in order to obtain error statistics for the orientation error and translation error.

5.2 Localization Results

Figures 17-21 plot the average translational error μ_t and standard deviation of the translational error σ_t, and the average orientation error μ_o and standard deviation of the orientation error σ_o for five fixed values of d. Tables 2-6 summarize these results. For the most part, Curve SLAM is more accurate than SPTAM and OKVIS in DS1, DS3, and DS4 for a number of reasons. First, these environments lacked distinguishable feature points (sample images of the data sets are shown in Figures 12–16). Second, Curve SLAM is not dependent on tracking large numbers of feature points between temporally successive stereo frames (for further discussion of Curve SLAM's dependence on tracking landmarks see Section 4.1). Third, SPTAM and OKVIS failed to track feature points robustly. Therefore, the ability of SPTAM and OKVIS to accurately localize and map these settings declined. In fact, at one point during DS3, OKVIS reported a tracking failure. In DS1, we were unable to apply SPTAM altogether because SPTAM failed to track feature points. It is also interesting to note that during short periods of operation in DS3 and DS4, SPTAM occasionally failed to track feature points correctly, causing major localization and mapping errors. In contrast, one reason that OKVIS operates in DS1, DS3, and DS4 better than SPTAM is because OKVIS relies on an IMU to localize its position. However, even in these settings, the performance of OKVIS diminishes.

DS2 contained better conditions for detecting, describing, and tracking feature points, thus for DS2, Curve SLAM is slightly less accurate than SPTAM and OKVIS. However, the maximum error estimates and attitude estimates provided by Curve SLAM are better than OKVIS and SPTAM over all distances traveled in DS2. In DS5, our algorithm is less accurate than OKVIS for a few reasons: DS5 contained better conditions for detecting, describing, and tracking feature points. Additionally, in DS5 we relied on a pre-trained convolutional neural network to detect the road. During short periods of operation, Curve SLAM was unable to detect the road, and our algorithm was forced to rely solely on propagating the IMU. Finally,

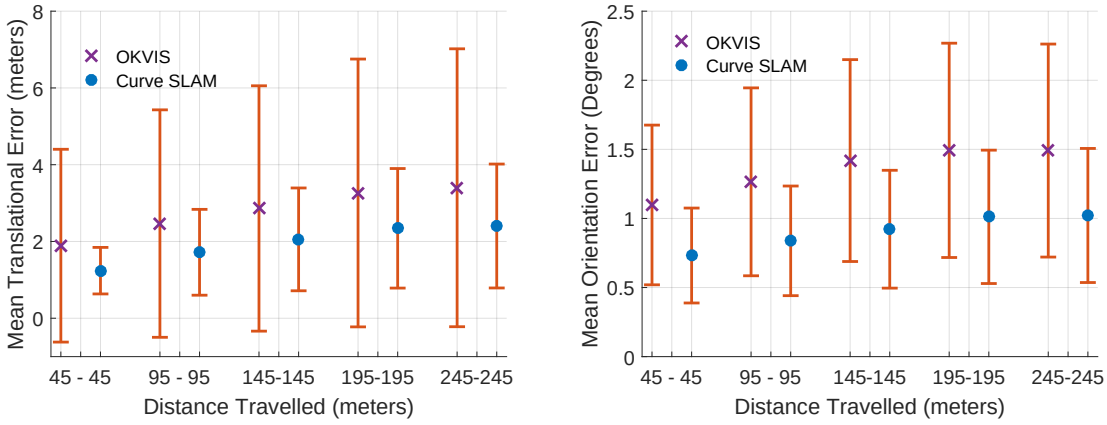


Figure 17: The average and standard deviation of the translational error and orientation error of Curve SLAM and OKVIS for DS1.

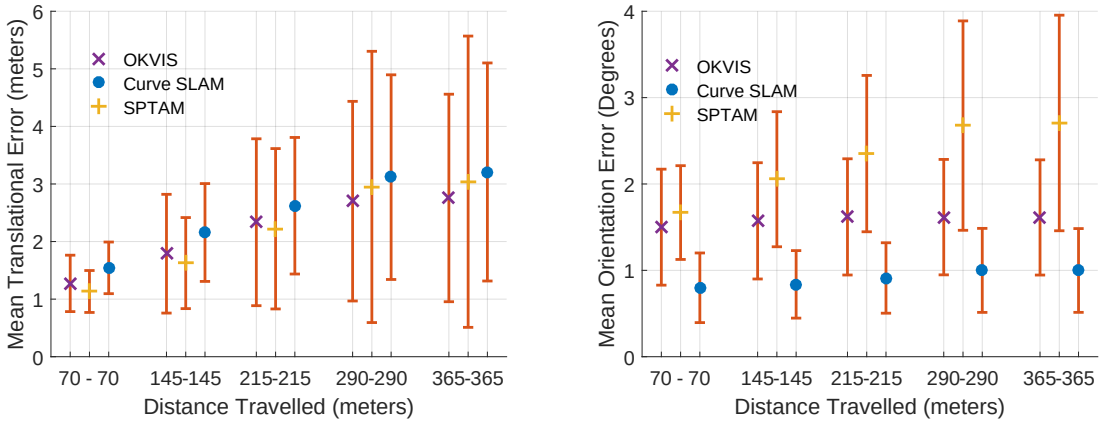


Figure 18: The average and standard deviation of the translational error and orientation error of Curve SLAM, SPTAM, and OKVIS for DS2.

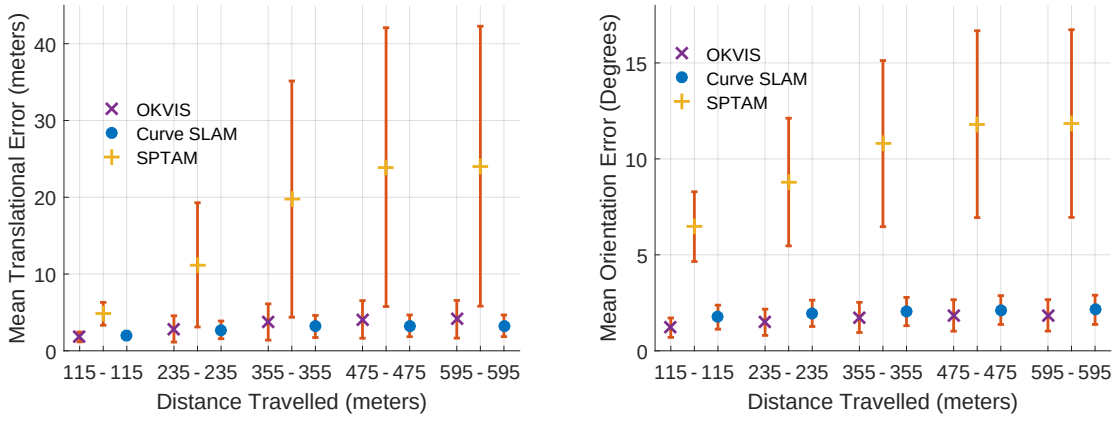


Figure 19: The average and standard deviation of the translational error and orientation error of Curve SLAM, SPTAM, and OKVIS for DS3.

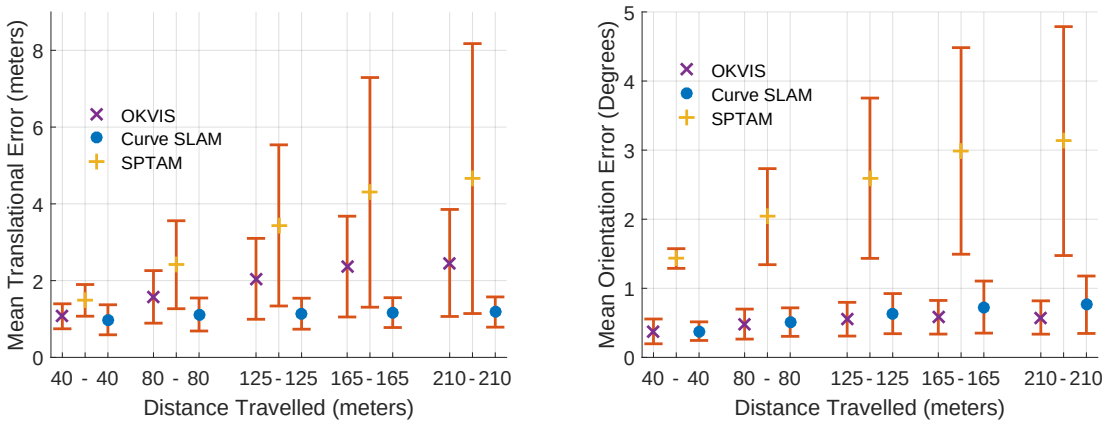


Figure 20: The average and standard deviation of the translational error and orientation error of Curve SLAM, SPTAM, and OKVIS for DS4.

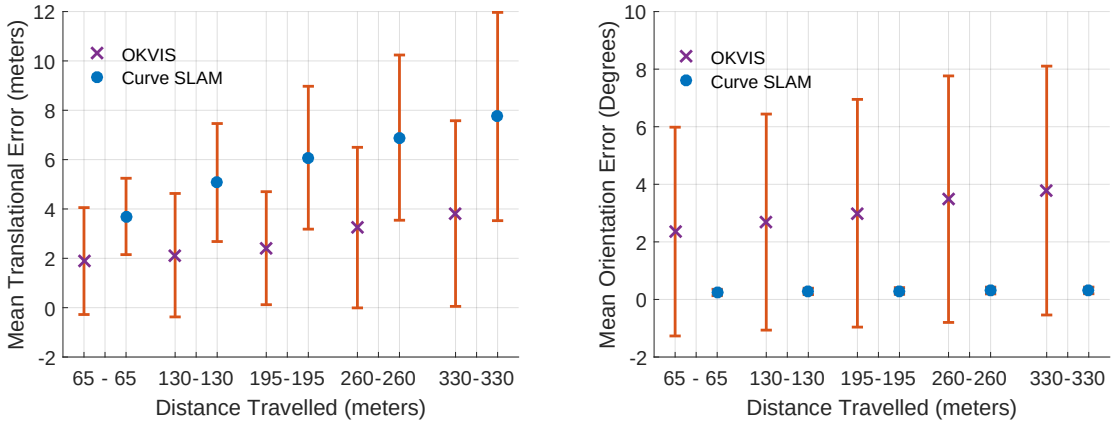


Figure 21: The average and standard deviation of the translational error and orientation error of Curve SLAM and OKVIS for DS5.

Table 2: Algorithm Comparison for DS1.

DS1 was collected at dawn under clear weather conditions. We did not apply SPTAM to DS1 because we were unable to track a sufficient number of feature points.

Paramater	SPTAM	Curve SLAM	OKVIS
Map Landmarks	NA	160 control points	279690
Distance Traveled: 45 Meters			
μ_t, μ_o	NA	1.24 m, 0.73°	1.89 m, 1.10°
σ_t, σ_o	NA	.61 m, 0.34°	2.51 m, 0.58°
Max error (trans., att.)	NA	2.73 m, 1.75°	10.61 m, 3.56°
μ_t/d	NA	2.8 %	4.2 %
Distance Traveled: 95 Meters			
μ_t, μ_o	NA	1.72 m, 0.84°	2.47 m, 1.27°
σ_t, σ_o	NA	1.12 m, 0.40°	2.96 m, 0.68°
Max error (trans., att.)	NA	5.43 m, 1.90°	10.98 m, 3.91°
μ_t/d	NA	1.81 %	2.6 %
Distance Traveled: 145 Meters			
μ_t, μ_o	NA	2.05 m, 0.92°	2.86 m, 1.42°
σ_t, σ_o	NA	1.34 m, 0.43°	3.2 m, 0.73°
Max error (trans., att.)	NA	6.91 m, 2.10°	11.55 m, 4.14°
μ_t/d	NA	1.42 %	2.0 %
Distance Traveled: 195 Meters			
μ_t, μ_o	NA	2.34 m, 1.01°	3.26 m, 1.49°
σ_t, σ_o	NA	1.56 m, 0.48°	3.49 m, 0.78°
Max error (trans., att.)	NA	6.91 m, 2.29°	12.35 m, 4.76°
μ_t/d	NA	1.2 %	1.67 %
Distance Traveled: 245 Meters			
μ_t, μ_o	NA	2.40 m, 1.01°	3.40 m, 1.49°
σ_t, σ_o	NA	1.61 m, 0.49°	3.62 m, 0.77°
Max error (trans., att.)	NA	6.91 m, 2.29°	12.73 m, 4.76°
μ_t/d	NA	.98 %	1.39 %

Table 3: Algorithm Comparison for DS2.

DS2 was collected at 2 PM with light rain and mostly sunny conditions.

Paramater	SPTAM	Curve SLAM	OKVIS
Map Landmarks	138403 feature points	220 control points	345340
Distance Traveled: 70 Meters			
μ_t, μ_o	1.13 m, 1.67°	1.54 m, 0.80°	1.27 m, 1.5°
σ_t, σ_o	0.36 m, 0.54°	0.45 m, 0.40°	0.49 m, 0.67°
Max error (trans., att.)	2.58 m, 3.29°	2.57 m, 1.86°	4.68 m, 4.97°
μ_t/d	1.62 %	2.2 %	1.82 %
Distance Traveled: 145 Meters			
μ_t, μ_o	1.63 m, 2.06°	2.15 m, 0.84°	1.79 m, 1.57°
σ_t, σ_o	0.79 m, 0.78°	0.85 m, 0.39°	1.03 m, 0.67°
Max error (trans., att.)	4.91 m, 4.19°	4.24 m, 1.89°	9.34 m, 4.97°
μ_t/d	1.12 %	1.5 %	1.23 %
Distance Traveled: 215 Meters			
μ_t, μ_o	2.22 m, 2.35°	2.62 m, 0.91°	2.34 m, 1.62°
σ_t, σ_o	1.39 m, 0.91°	1.19 m, 0.41°	1.45 m, 0.67°
Max error (trans., att.)	8.41 m, 4.87°	6.12 m, 2.06°	10.51 m, 4.97°
μ_t/d	1.03 %	1.22 %	1.1 %
Distance Traveled: 290 Meters			
μ_t, μ_o	2.95 m, 2.68°	3.12 m, 1.00°	2.70 m, 1.62°
σ_t, σ_o	2.36 m, 1.21°	1.78 m, 0.49°	1.73 m, .67°
Max error (trans., att.)	12.58 m, 6.78°	9.07 m, 2.76°	14.71 m, 4.97°
μ_t/d	1.02 %	0.34 %	0.93 %
Distance Traveled: 365 Meters			
μ_t, μ_o	3.04 m, 2.71°	3.21 m, 1.00°	2.76 m, 1.61°
σ_t, σ_o	2.53 m, 1.25°	1.89 m, 0.49°	1.80 m, 0.67°
Max error (trans., att.)	15.24 m, 6.78°	9.64 m, 2.76°	19.44 m, 4.97°
μ_t/d	.83 %	0.88 %	0.76 %

Table 4: Algorithm Comparison for DS3.

DS3 was collected near sunset under mostly sunny conditions.

Paramater	SPTAM	Curve SLAM	OKVIS
Map Landmarks	151867 feature points	348 control points	495874
Distance Traveled: 115 Meters			
μ_t, μ_o	4.82 m, 6.47°	1.99 m, 1.75°	1.80 m, 1.20°
σ_t, σ_o	1.49 m, 1.82°	0.52 m, 0.63°	0.63 m, 0.51°
Max error (trans., att.)	22.27 m, 29.44°	3.19 m, 3.47°	6.57 m, 4.02°
μ_t/d	4.19 %	1.73 %	1.57 %
Distance Traveled: 235 Meters			
μ_t, μ_o	11.19 m, 8.79°	2.73 m, 1.95°	2.85 m, 1.49°
σ_t, σ_o	8.09 m, 3.33°	1.15 m, 0.69°	1.71 m, 0.68°
Max error (trans., att.)	49.77 m, 33.60°	5.63 m, 3.87°	12.62 m, 4.12°
μ_t/d	4.76 %	1.16 %	1.21 %
Distance Traveled: 355 Meters			
μ_t, μ_o	19.76 m, 10.79°	3.17 m, 2.04°	3.75 m, 1.74°
σ_t, σ_o	15.39 m, 4.33°	1.44 m, 0.74°	2.36 m, 0.79°
Max error (trans., att.)	54.36 m, 33.60°	7.80 m, 3.87°	12.62 m, 4.12°
μ_t/d	5.57 %	0.89 %	1.06 %
Distance Traveled: 475 Meters			
μ_t, μ_o	23.93 m, 11.81°	3.25 m, 2.12°	4.08 m, 1.84°
σ_t, σ_o	18.16 m, 4.87°	1.42 m, 0.75°	2.45 m, 0.82°
Max error (trans., att.)	55.86 m, 33.60°	7.80 m, 3.87°	12.62 m, 4.12°
μ_t/d	5.04 %	0.68 %	0.86 %
Distance Traveled: 595 Meters			
μ_t, μ_o	24.05 m, 11.84°	3.25 m, 2.14°	4.11 m, 1.85°
σ_t, σ_o	18.23 m, 4.90°	1.41 m, 0.76°	2.46 m, 0.82°
Max error (trans., att.)	55.90 m, 33.60°	7.80 m, 3.97°	12.62 m, 4.12°
μ_t/d	4.04 %	0.55 %	0.69 %

Table 5: Algorithm Comparison for DS4.

DS4 was collected at 10 AM under mostly cloudy conditions and clear weather.

Paramater	SPTAM	Curve SLAM	OKVIS
Map Landmarks	55577 feature points	92 control points	17805
Distance Traveled: 40 Meters			
μ_t, μ_o	1.49 m, 1.43°	0.98 m, 0.38°	1.07 m, 0.38°
σ_t, σ_o	.41 m, 0.14°	0.39 m, 0.13°	0.33 m, 0.18°
Max error (trans., att.)	2.09 m, 1.69°	1.74 m, 0.74°	1.55 m, 0.97°
μ_t/d	3.71 %	2.45 %	2.70 %
Distance Traveled: 80 Meters			
μ_t, μ_o	2.41 m, 2.04°	1.12 m, 0.51°	1.58 m, 0.48°
σ_t, σ_o	1.15 m, 0.70°	0.43 m, 0.21°	0.68 m, 0.22°
Max error (trans., att.)	4.57 m, 3.07°	2.35 m, 1.03°	3.04 m, 1.00°
μ_t/d	3.02 %	1.40 %	1.97 %
Distance Traveled: 125 Meters			
μ_t, μ_o	3.44 m, 2.59°	1.14 m, 0.63°	2.05 m, 0.55°
σ_t, σ_o	2.10 m, 1.16°	.40 m, 0.29°	1.05 m, 0.24°
Max error (trans., att.)	7.69 m, 4.61°	2.35 m, 1.30°	4.19 m, 1.39°
μ_t/d	2.75 %	0.91 %	1.64 %
Distance Traveled: 165 Meters			
μ_t, μ_o	4.30 m, 2.99°	1.17 m, 0.73°	2.37 m, 0.58°
σ_t, σ_o	2.99 m, 1.50°	0.39 m, 0.38°	1.31 m, 0.24°
Max error (trans., att.)	11.23 m, 5.83°	2.35 m, 1.76°	5.27 m, 1.39°
μ_t/d	2.61 %	0.71 %	1.43 %
Distance Traveled: 210 Meters			
μ_t, μ_o	4.66 m, 11.84°	1.18 m, 0.76°	2.46 m, 0.58°
σ_t, σ_o	3.52 m, 4.90°	0.39 m, 0.42°	1.39 m, 0.24°
Max error (trans., att.)	15.33 m, 33.60°	2.35 m, 1.95°	5.92 m, 1.39°
μ_t/d	2.22 %	0.56 %	1.17 %

Table 6: Algorithm Comparison for DS5.

DS5 is a sequence obtained from the KITTI data set [Geiger et al., 2013].
DS5 was collected under sunny conditions.

Paramater	SPTAM	Curve SLAM	OKVIS
Map Landmarks	NA	96 control points	21402
Distance Traveled: 65 Meters			
μ_t, μ_o	NA	3.70 m, 0.24°	1.89 m, 2.36°
σ_t, σ_o	NA	1.55 m, 0.11°	2.17 m, 3.63°
Max error (trans., att.)	NA	7.15 m, 0.45°	10.37 m, 13.36°
μ_t/d	NA	5.69 %	2.91 %
Distance Traveled: 130 Meters			
μ_t, μ_o	NA	5.07 m, 0.28°	2.13 m, 2.69°
σ_t, σ_o	NA	2.39 m, 0.11°	2.50 m, 3.75°
Max error (trans., att.)	NA	11.62 m, 0.54°	15.59 m, 14.21°
μ_t/d	NA	3.90 %	1.64 %
Distance Traveled: 195 Meters			
μ_t, μ_o	NA	6.08 m, 0.29°	2.41 m, 2.99°
σ_t, σ_o	NA	2.90 m, 0.12°	2.29 m, 3.96°
Max error (trans., att.)	NA	14.02 m, 0.55°	15.59 m, 14.61°
μ_t/d	NA	3.12 %	1.24 %
Distance Traveled: 260 Meters			
μ_t, μ_o	NA	6.89 m, 0.31°	3.24 m, 3.48°
σ_t, σ_o	NA	3.35 m, 0.12°	3.25 m, 4.28°
Max error (trans., att.)	NA	15.6 m, 0.55°	15.59 m, 14.72°
μ_t/d	NA	2.65 %	1.25 %
Distance Traveled: 330 Meters			
μ_t, μ_o	NA	7.75 m, 0.31°	3.81 m, 3.78°
σ_t, σ_o	NA	4.22 m, 0.12°	3.76 m, 4.32°
Max error (trans., att.)	NA	18.54 m, 0.55°	16.79 m, 14.72°
μ_t/d	NA	2.35 %	1.16 %

in DS5, the camera to road distance was large, causing larger inaccuracies in our stereo triangulation method. However, in settings similar to DS2 and DS5 where large numbers of feature points are readily available for tracking, we expect a robust point-based feature detector/extractor tracking algorithm to provide a better motion estimate than Curve SLAM which tracks a limited number of landmark curves between temporally sequential stereo frames.

Figures 22–26 plot the body-frame velocity estimates and the attitude estimates of Curve SLAM, OKVIS, SPTAM, and the corresponding ground truth as function of time. The body frame velocity estimate for SPTAM was obtained by subtracting two world frame position estimates between chronologically successive image frames and multiplying by the frame rate to obtain a world frame velocity estimate, this world frame velocity estimate was then transformed to the body frame using SPTAM's estimated attitude. In all five data sets, Curve SLAM's attitude estimate is accurate without relying on an external compass. It should also be noted that without the vision-based curve measurements, the IMU alone would produce quickly growing attitude error estimates.

Figures 28–32 plot the estimated camera trajectory of Curve SLAM, OKVIS, SPTAM, and the corresponding ground truth trajectory using google maps for all five data sets. To overlay the estimated trajectory onto google maps, we align the frame where the GPS ground truth coordinates are defined with the world frame of each of the data sets. To align coordinate frames, we must find a coordinate transformation that maps a point $\mathbf{r}_k^{\mathcal{W}}$ defined in the coordinates of the world frame $\mathcal{W}$ (the world frame represents the coordinate frame in which each data set is defined) to a point $\mathbf{r}_k^{\mathcal{W}'}$ defined in the coordinates of the GPS frame $\mathcal{W}'$, where $\mathbf{r}_k^{\mathcal{W}}$ represents the estimated location of the sensor platform in the world frame at time step k. The coordinate transformation that aligns the world frame with the GPS frame is expressed as

$$\mathbf{r}_k^{\mathcal{W}'} = \mathbf{R}_{\mathcal{W}}^{\mathcal{W}'} \mathbf{r}_k^{\mathcal{W}} + \mathbf{c}_{\mathcal{W}}^{\mathcal{W}'} \tag{10}$$

The matrix $\mathbf{R}_{\mathcal{W}}^{\mathcal{W}'}$ and translation vector $\mathbf{c}_{\mathcal{W}}^{\mathcal{W}'}$ are obtained by minimizing $\sum_k ||\mathbf{r}_k^{\mathcal{W}'} - \mathbf{g}_k^{\mathcal{W}'}||^2$ where $\mathbf{g}_k^{\mathcal{W}'}$ represents ground truth coordinates.

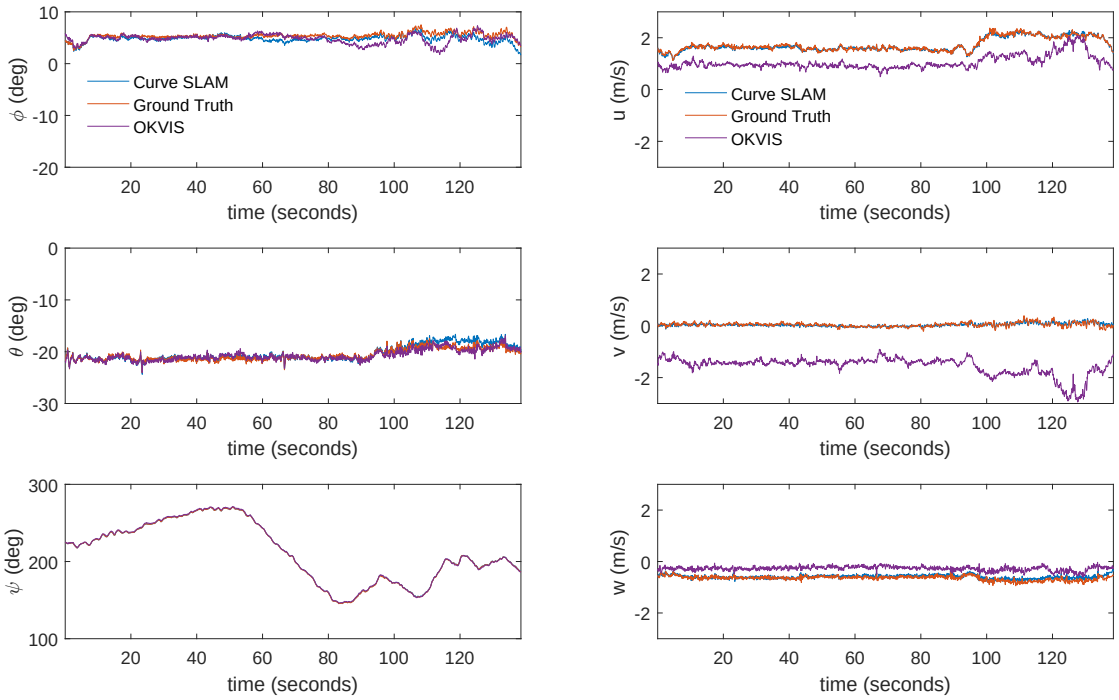


Figure 22: Attitude and body-frame velocity estimates of Curve SLAM along with the corresponding ground truth for DS1.

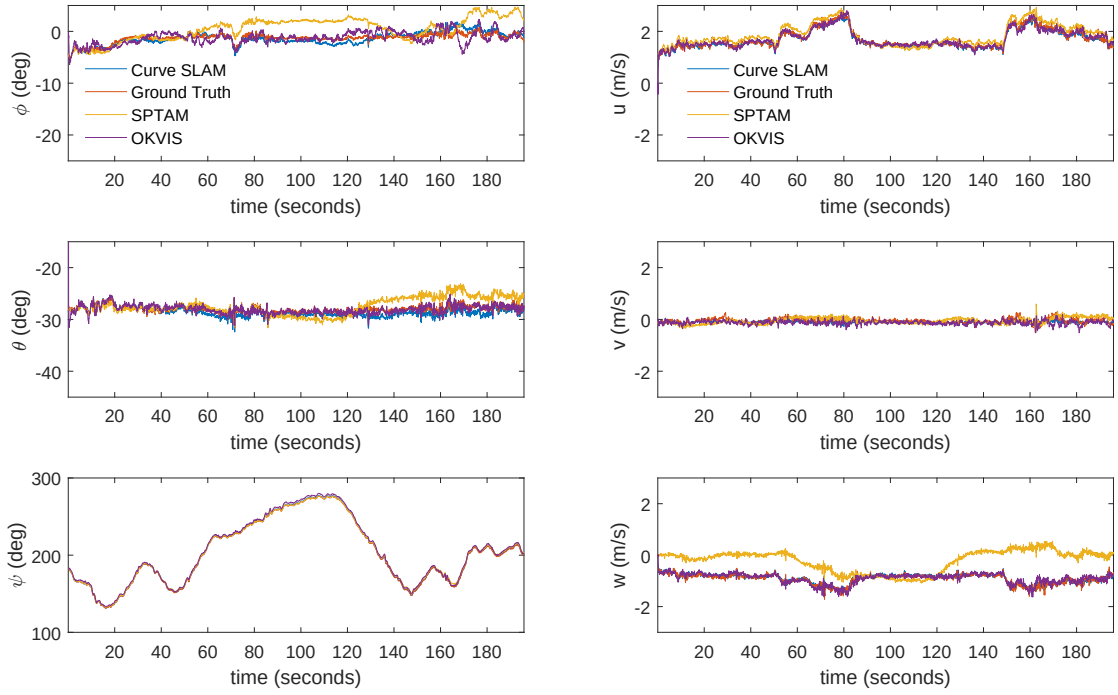


Figure 23: Attitude and body-frame velocity estimates of Curve SLAM and SPTAM along with the corresponding ground truth for DS2.

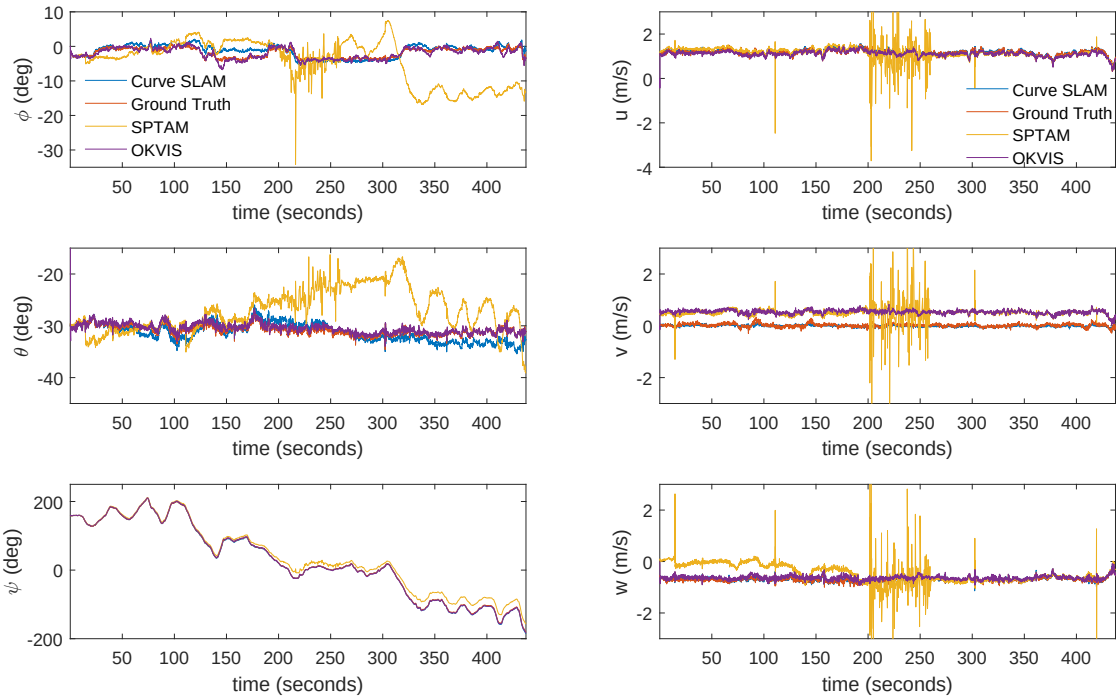


Figure 24: Attitude and body-frame velocity estimates of Curve SLAM and SPTAM along with the corresponding ground truth for DS3.

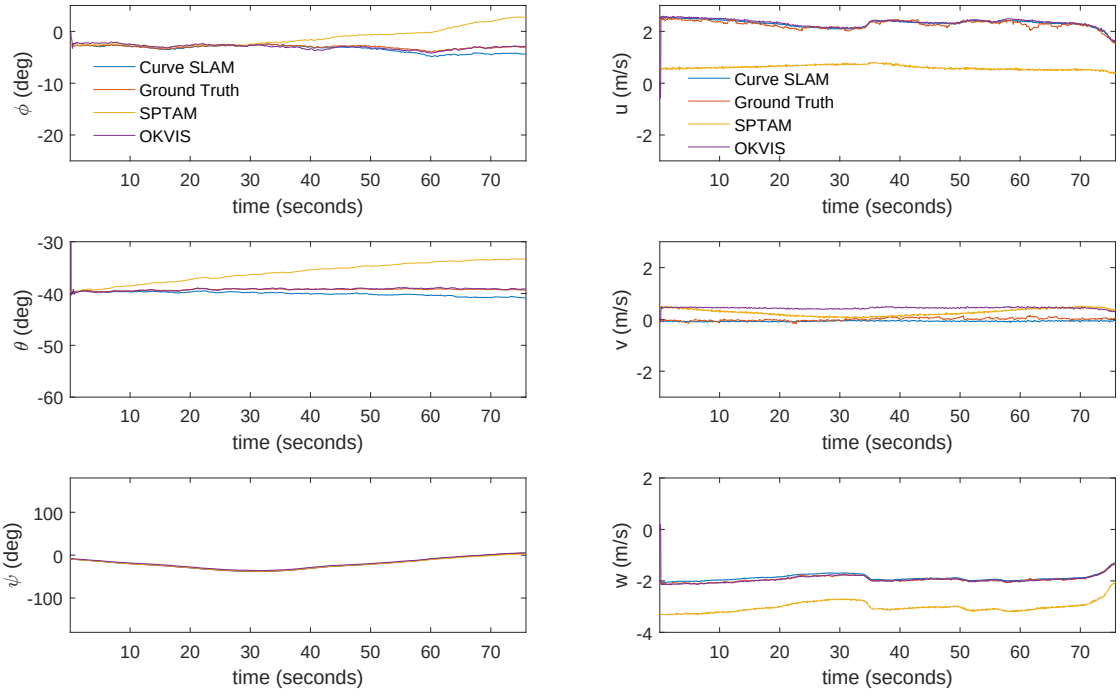


Figure 25: Attitude and body-frame velocity estimates of Curve SLAM and SPTAM along with the corresponding ground truth for DS4.

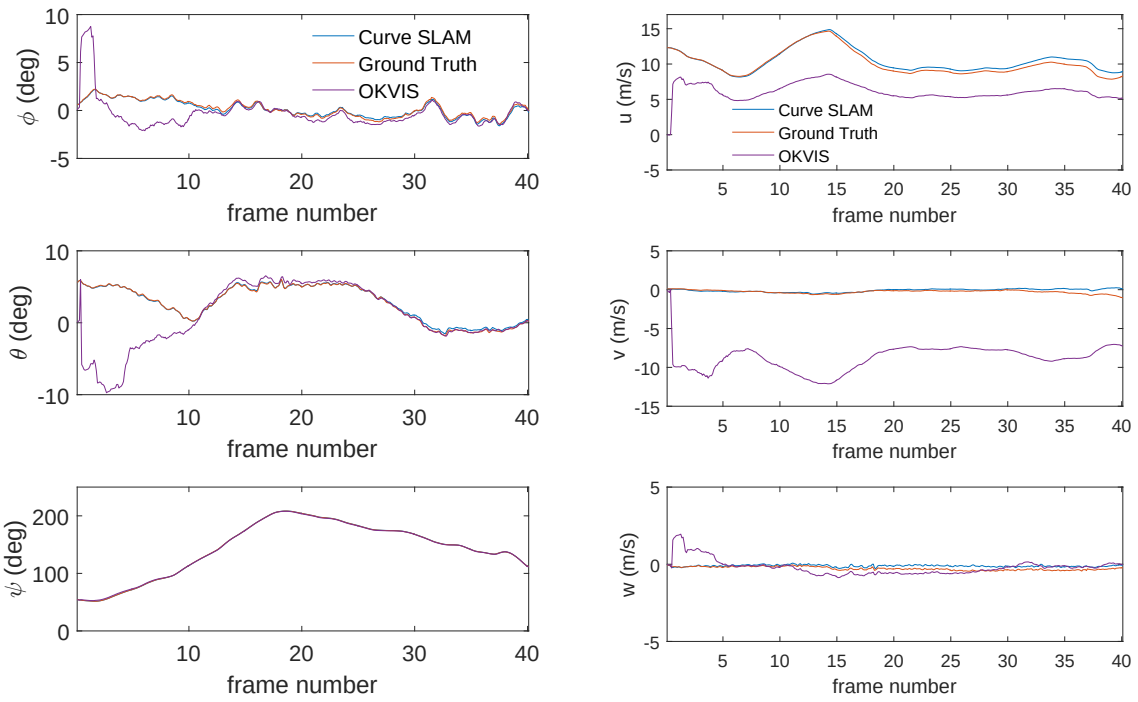


Figure 26: Attitude and body-frame velocity estimates of Curve SLAM and SPTAM along with the corresponding ground truth for DS5.

In Figures 28, 29, and 30 the labeled white curve is the sidewalk path that was used to provide curves in the Curve SLAM algorithm. In Figure 31, the center curve is a road and yellow road lanes located on this road were used as curves in the Curve SLAM algorithm. In Figure 32, the labeled road provided curves for the Curve SLAM algorithm. In DS3, between the start point and end point of the loop, Curve SLAM estimated the magnitude error between the start pose and end pose to be 6.4 meters, OKVIS estimated the magnitude error between the start pose and end pose to be 7.1 meters. By comparision, ground-truth recorded a magnitude error of 4.24 meters between the start pose and end pose. At the final pose, the final attitude error between ground truth and Curve SLAM is 3.8 degrees in yaw, 1.61 degrees in pitch and 2.12 degrees in roll. The final attitude error between OKVIS and ground truth is 1.94 degrees in yaw, .01 degrees in pitch and .04 degrees in roll.

5.3 Mapping Results

For mapping purposes, we further reduce the number of points required to represent the path using the method described in Section 4.8. A plot of these results is shown in Figure 33 for DS1, Figure 34 for DS2, Figure 35 for DS3, Figure 36 for DS4, and Figure 37 for DS5. The mapping results are obtained by transforming the map curves from the world frame where each data set is defined to the coordinate frame where GPS is defined. The transformation used in this process is given by (10). Figure 27 displays the number of landmarks required to map the five data sets for Curve SLAM and SPTAM. In all the applicable data sets, Curve SLAM requires roughly three orders of magnitude fewer landmarks to represent the map (see Tables 2-6). Indeed, in DS1 only 160 control points are used to represent 252 meters of path, in DS2 only 220 control points are used to represent 375 meters of a path, in DS3 only 348 control points are used to represent 603 meters of a path, in DS4 only 92 control points are used to represent 230 meters of path, and in DS5 only 96 control points are used to represent 435 meters of path.

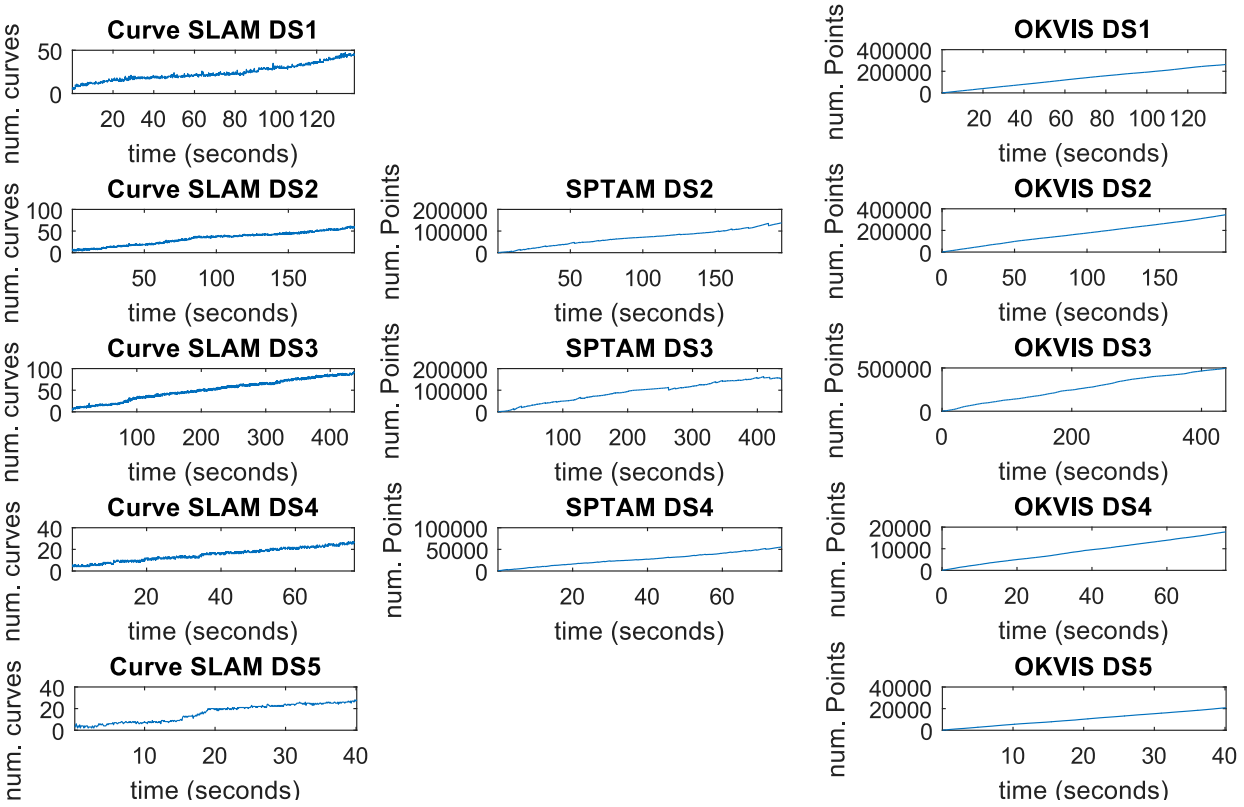


Figure 27: The number of curves (Curve SLAM) and points (SPTAM) required to represent the map as a function of time for the five data sets.

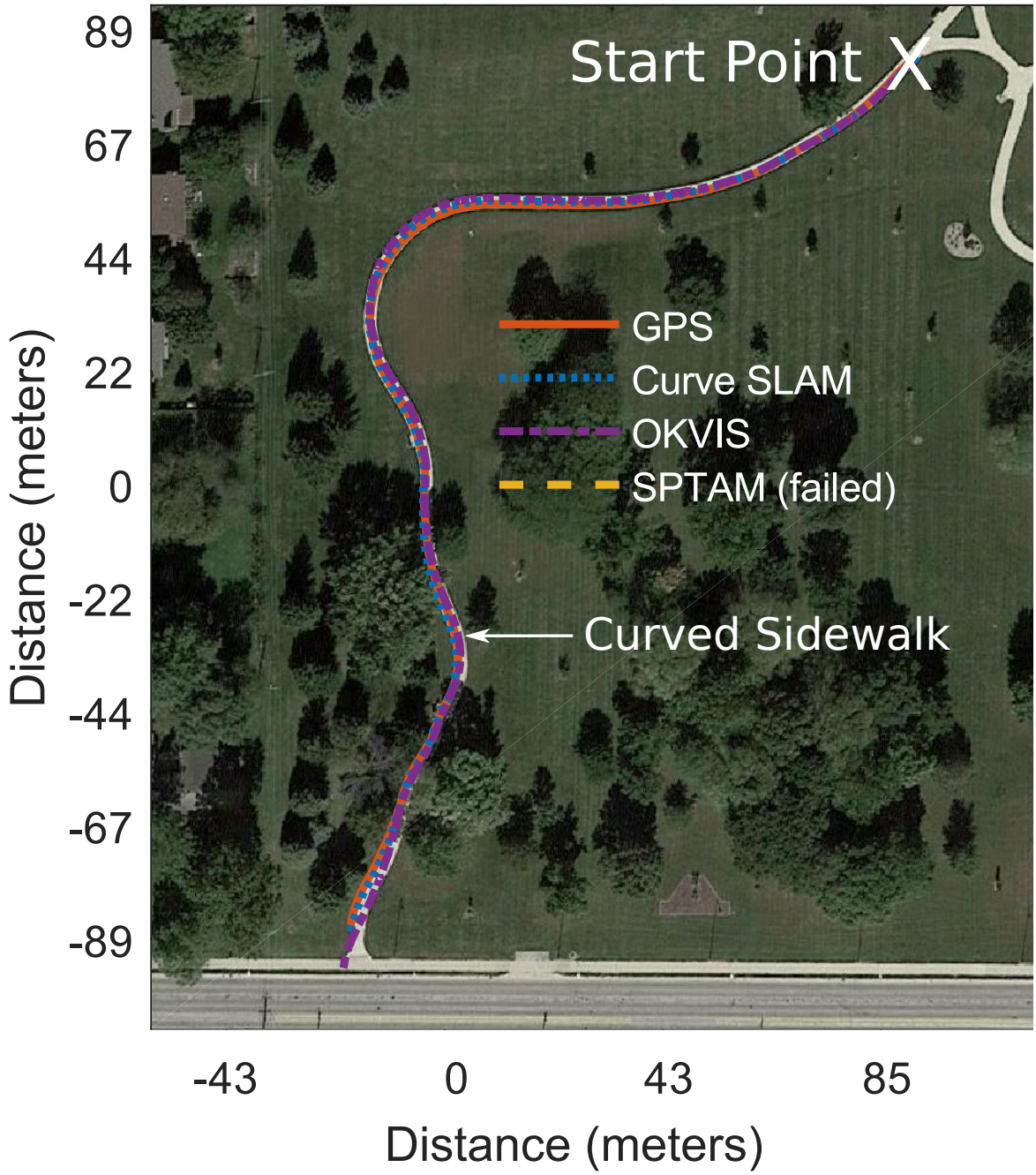


Figure 28: Localization results plotted in google maps for DS1. The estimated trajectory of Curve SLAM is plotted in dashed blue, the estimated trajectory of OKVIS is plotted in dashed purple, and the ground truth trajectory of the stereo camera is plotted in red. The estimated trajectory of SPTAM is not visible because the setting lacks distinguishable feature points, and SPTAM failed to track a sufficient number of feature points in this environment. The start point of the GPS track is marked with an X. DS1 was collected at dawn under clear weather conditions. The experiment lasted roughly 138.5 seconds. During this time, our sensors traveled approximately 252.2 meters.

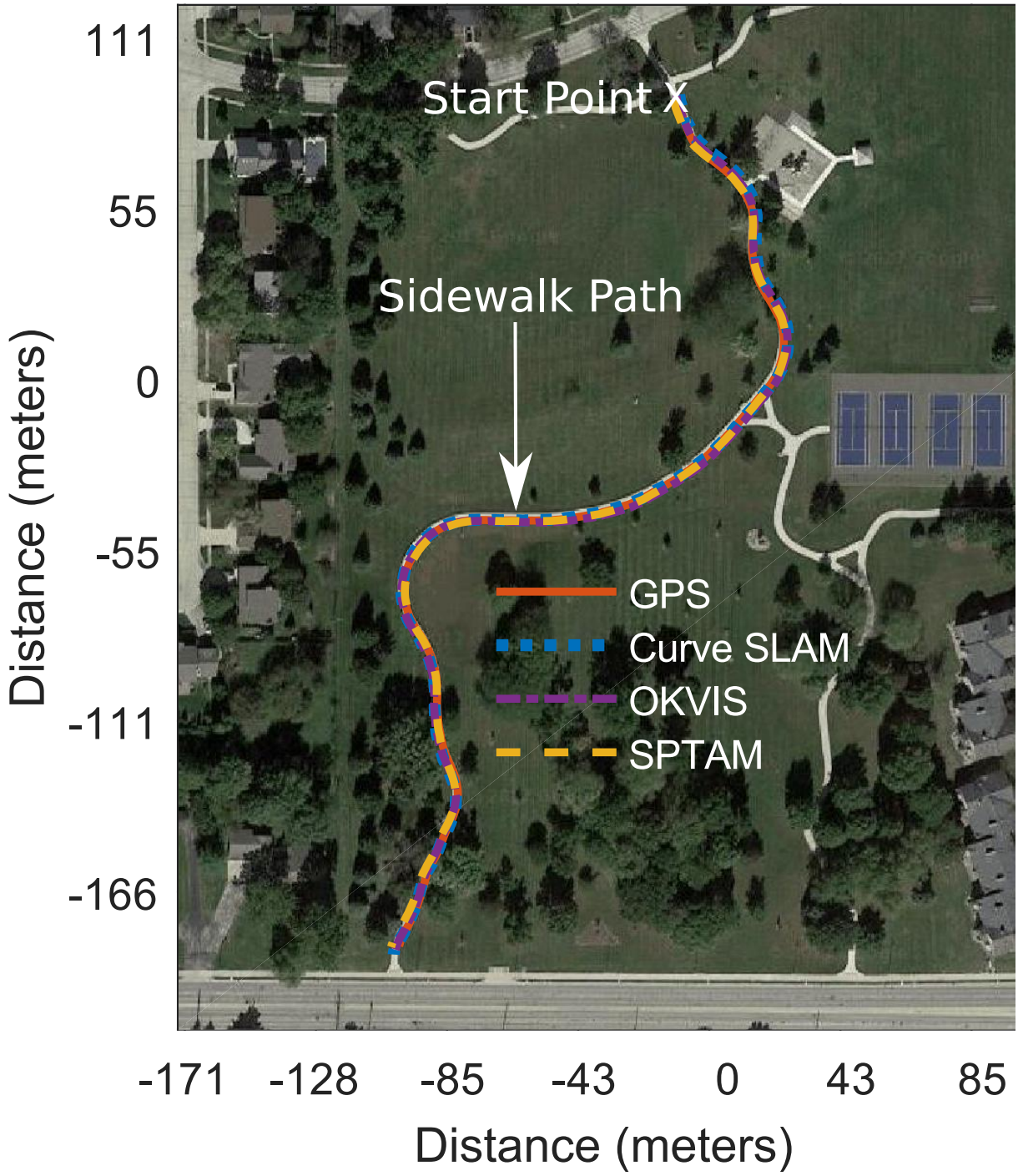


Figure 29: Localization results plotted in google maps for DS2. The estimated trajectory of Curve SLAM is plotted in dashed blue, the estimated trajectory of OKVIS is plotted in dashed purple, the estimated trajectory of SPTAM is plotted in dashed yellow, and the ground truth trajectory of the stereo camera is plotted in red. The start point of the GPS track is marked with an X. DS2 was collected in the mid-afternoon, roughly 2 PM. Part of DS2 was collected during a light rainstorm under cloudy conditions, while the other part of DS2 was collected immediately following this light rainstorm under mostly sunny conditions. DS2 lasted roughly 195.85 seconds. During this time, our sensors traveled approximately 375 meters.

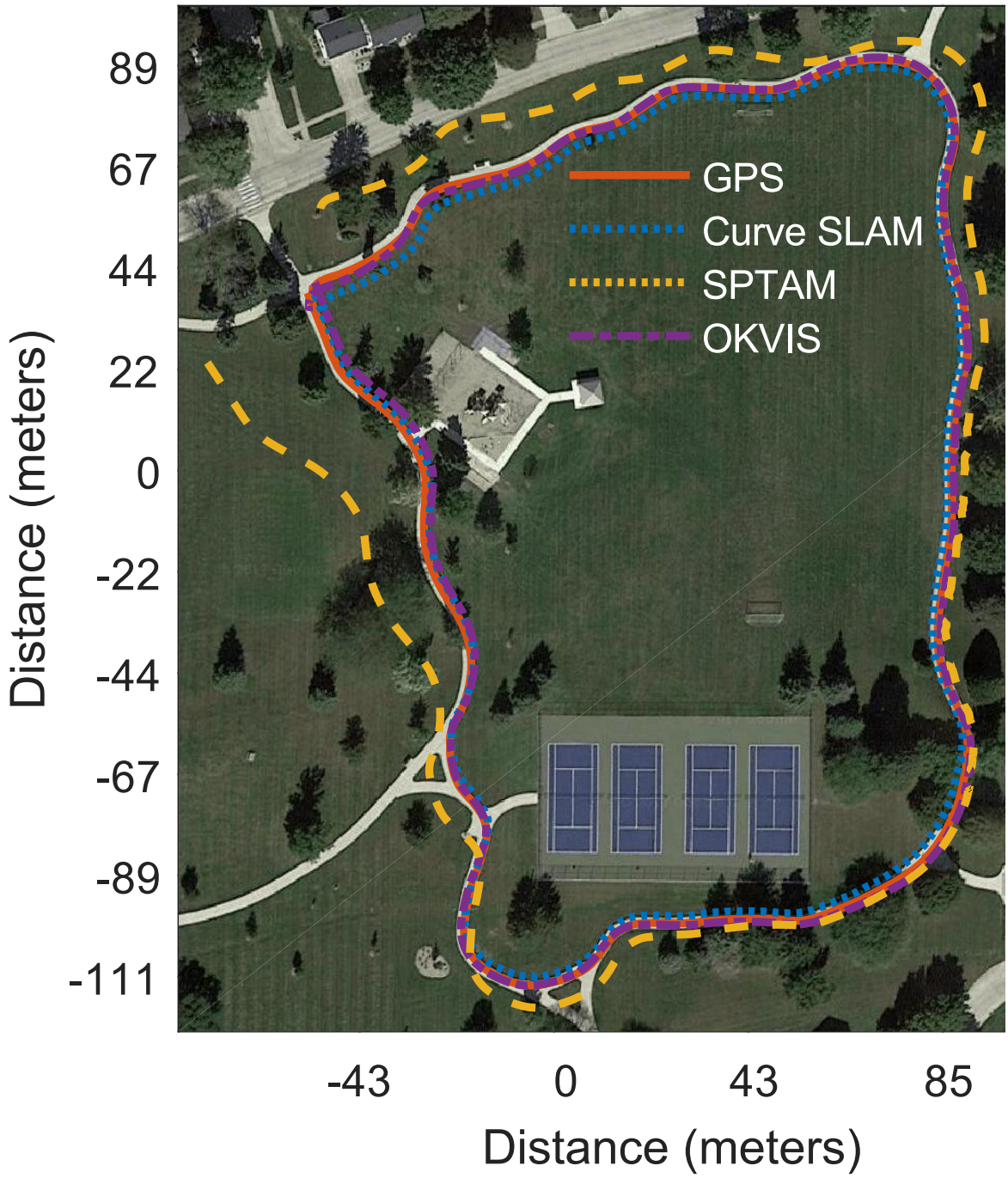


Figure 30: Localization results plotted in google maps for DS3. The estimated trajectory of Curve SLAM is plotted in dashed blue, the estimated trajectory of OKVIS is plotted in dashed purple, the estimated trajectory of SPTAM is plotted in dashed yellow, and the ground truth trajectory of the stereo camera is plotted in red. The start point of the GPS track is marked with an X. DS3 was collected about an hour prior to sunset on a mostly sunny day with clear weather conditions. DS3 lasted roughly 437.55 seconds. During this time, our sensors traveled approximately 603.4 meters.

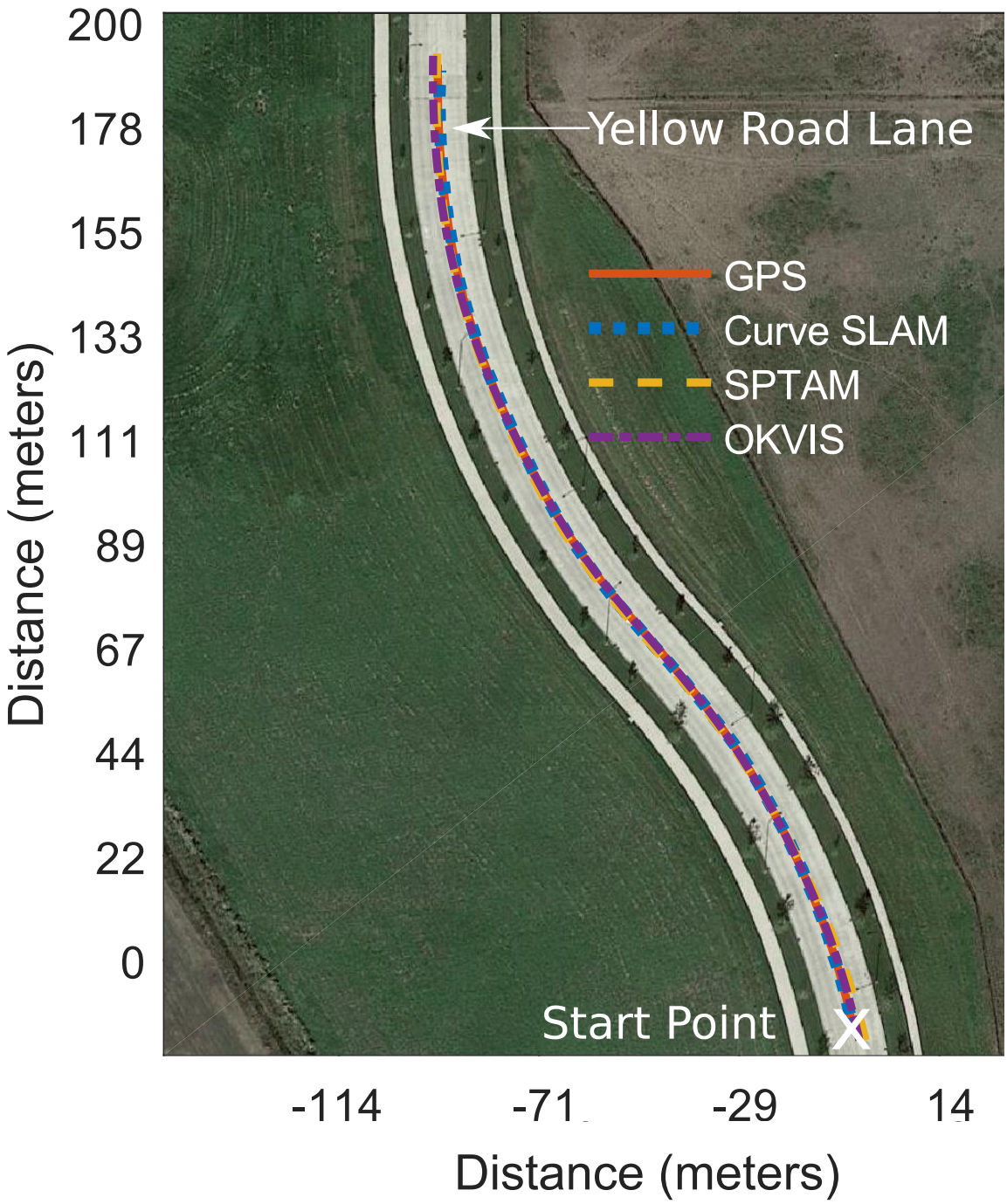


Figure 31: Localization results plotted in google maps for DS4. The estimated trajectory of Curve SLAM is plotted in dashed blue, the estimated trajectory of OKVIS is plotted in dashed purple, the estimated trajectory of SPTAM is plotted in dashed yellow, and the ground truth trajectory of the stereo camera is plotted in red. The start point of the GPS track is marked with an X. DS4 was collected in the morning hours under mostly cloudy conditions with clear weather. DS4 lasted roughly 70 seconds. During this time, our sensors traveled approximately 230 meters.

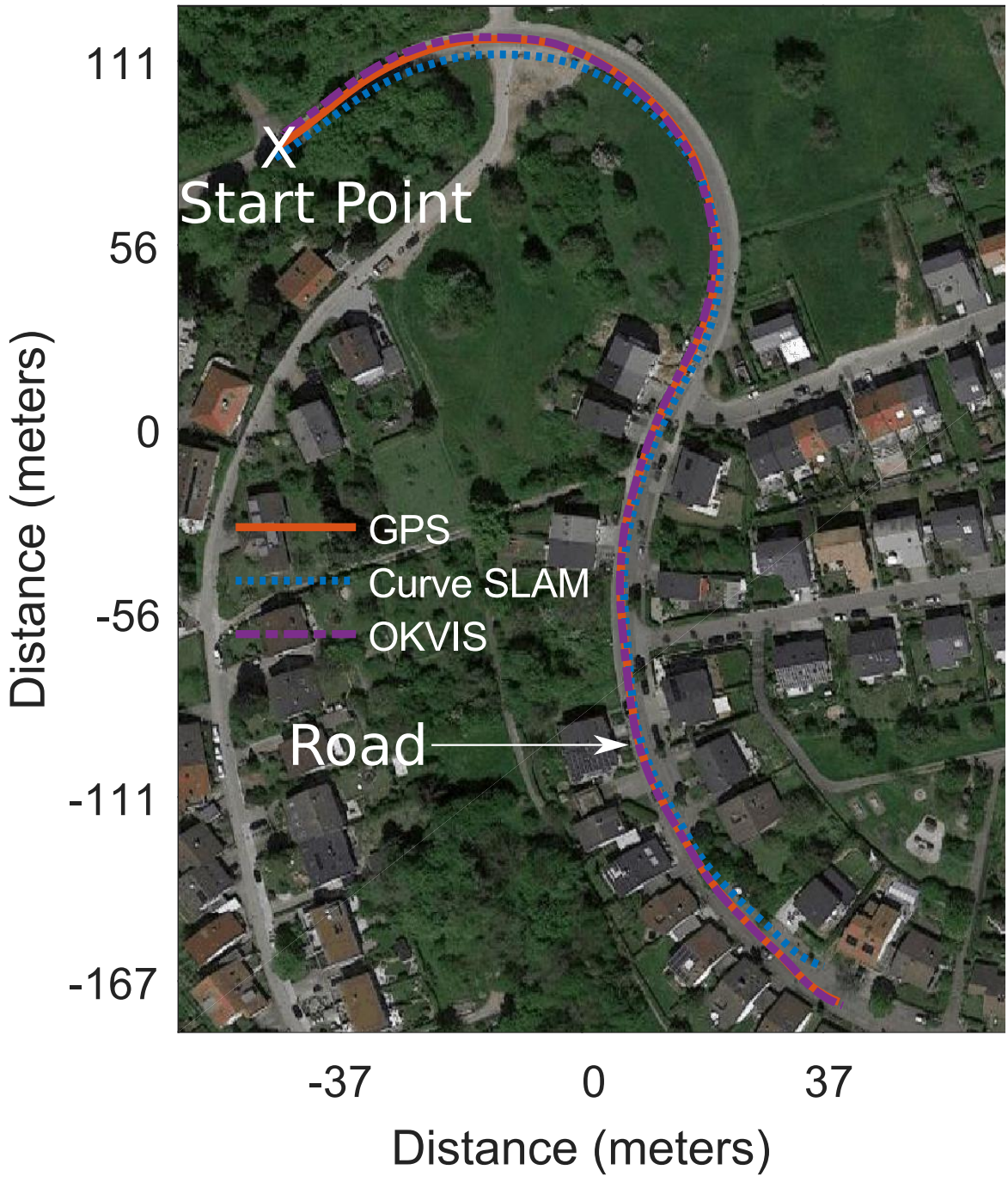


Figure 32: Localization results plotted in google maps for DS5. The estimated trajectory of Curve SLAM is plotted in dashed blue, the estimated trajectory of OKVIS is plotted in dashed purple, and the ground truth trajectory of the stereo camera is plotted in red. The start point of the GPS track is marked with an X. DS5 contains a sequence of data obtained from the KITTI data set, and was collected under sunny conditions. This sequence was selected due to the presence of curves in the environment and the number of occlusions covering the road.

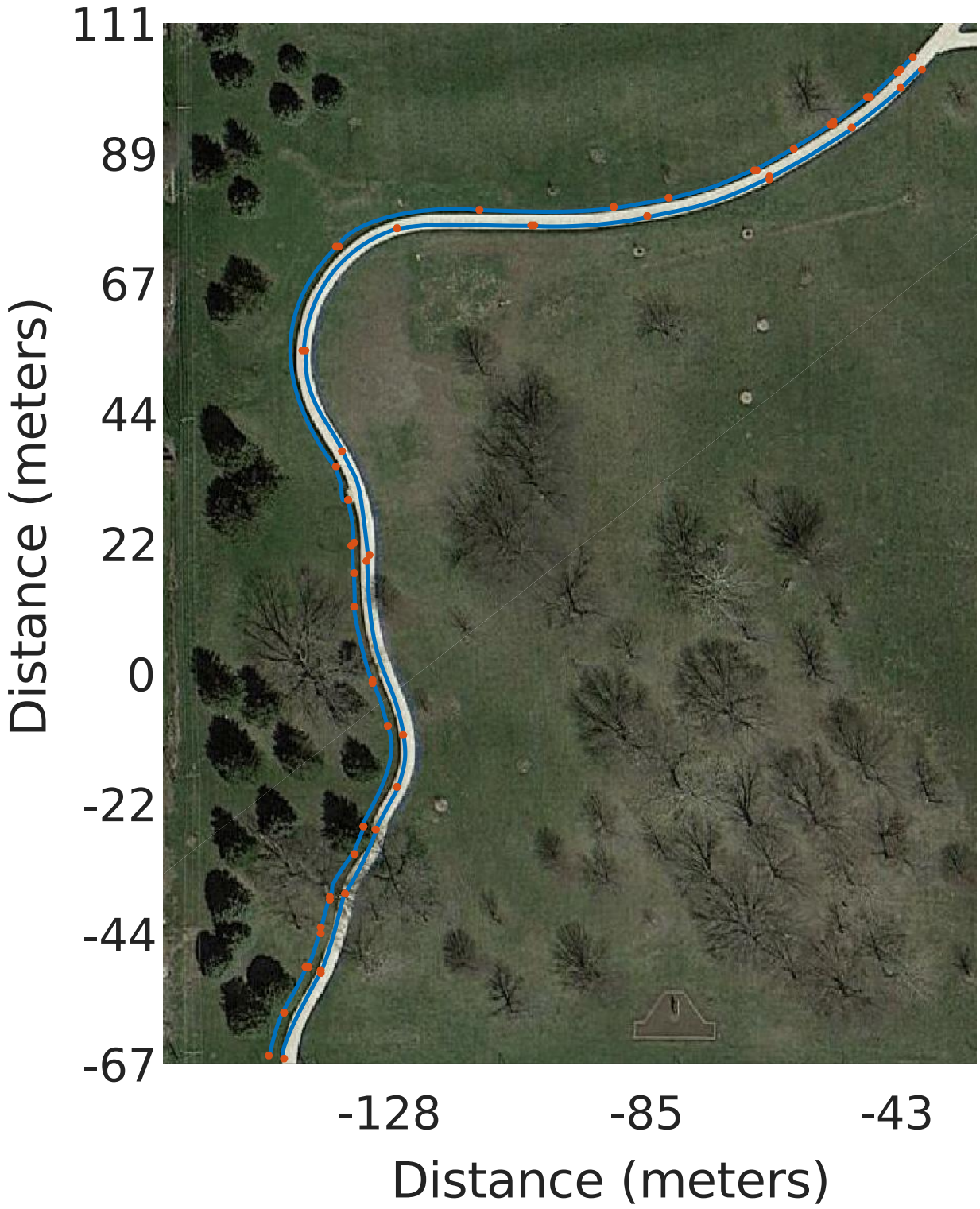


Figure 33: Mapping results plotted in google maps for DS1. The mapping results display the reduced number of curves required to estimate the sidewalk. Red points denote the start and end points of curve segments, The blue plot represents the location of curves interpolated with the estimated control points.

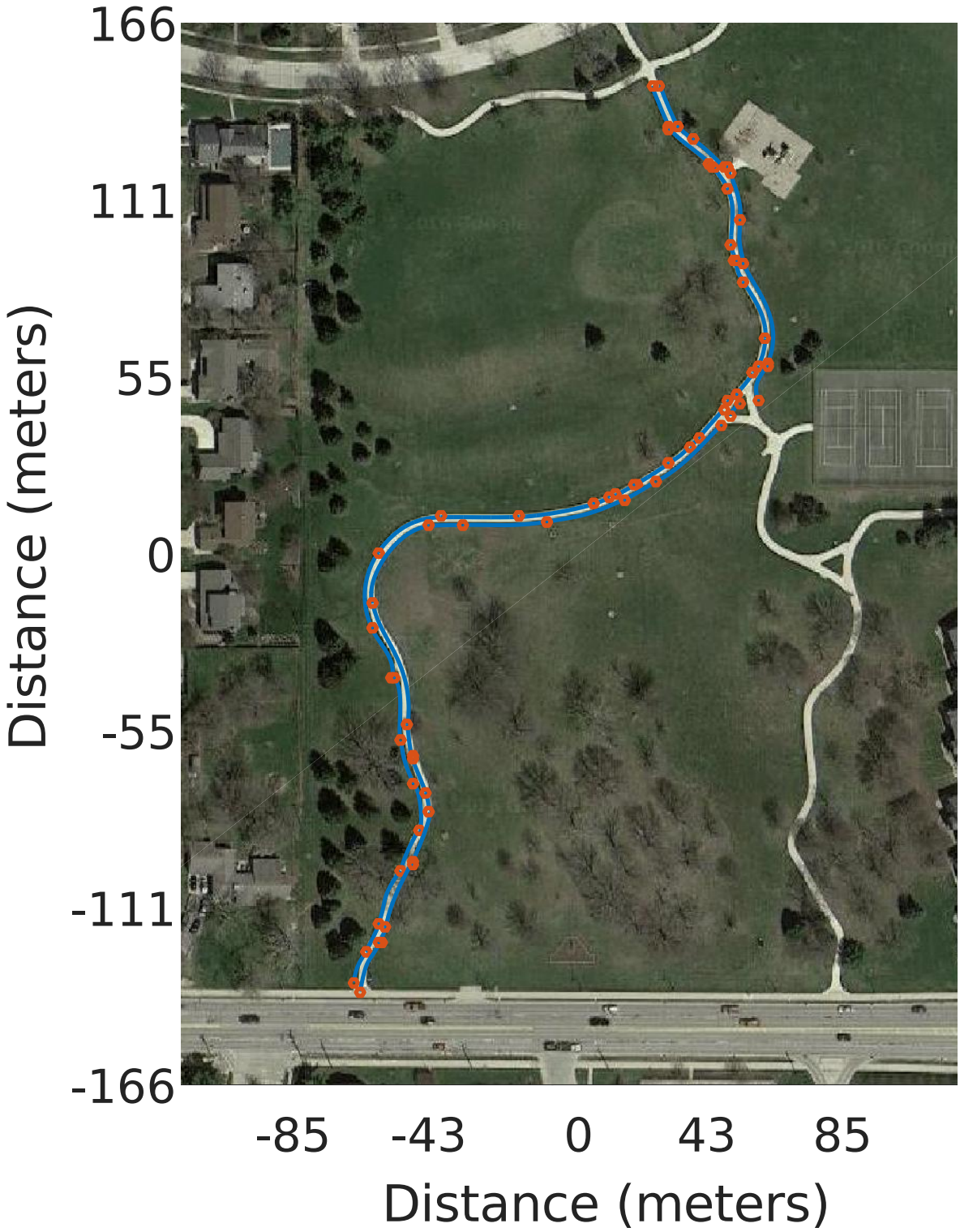


Figure 34: Mapping results plotted in google maps for DS2. The mapping results display the reduced number of curves required to estimate the sidewalk. Red points denote the start and end points of curve segments, The blue plot represents the location of curves interpolated with the estimated control points.

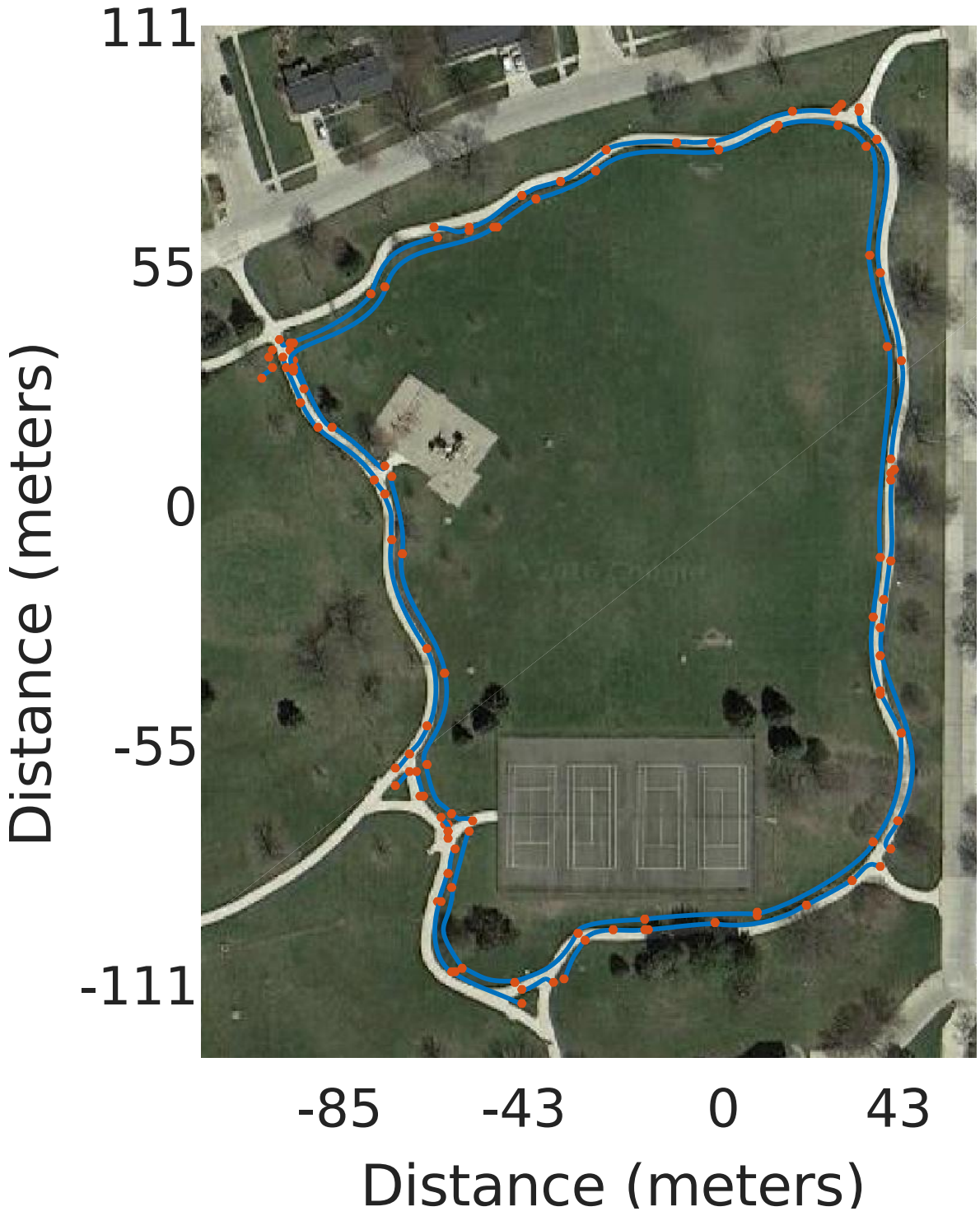


Figure 35: Mapping results plotted in google maps for DS3. The mapping results display the reduced number of curves required to estimate the sidewalk. Red points denote the start and end points of curve segments, The blue plot represents the location of curves interpolated with the estimated control points.

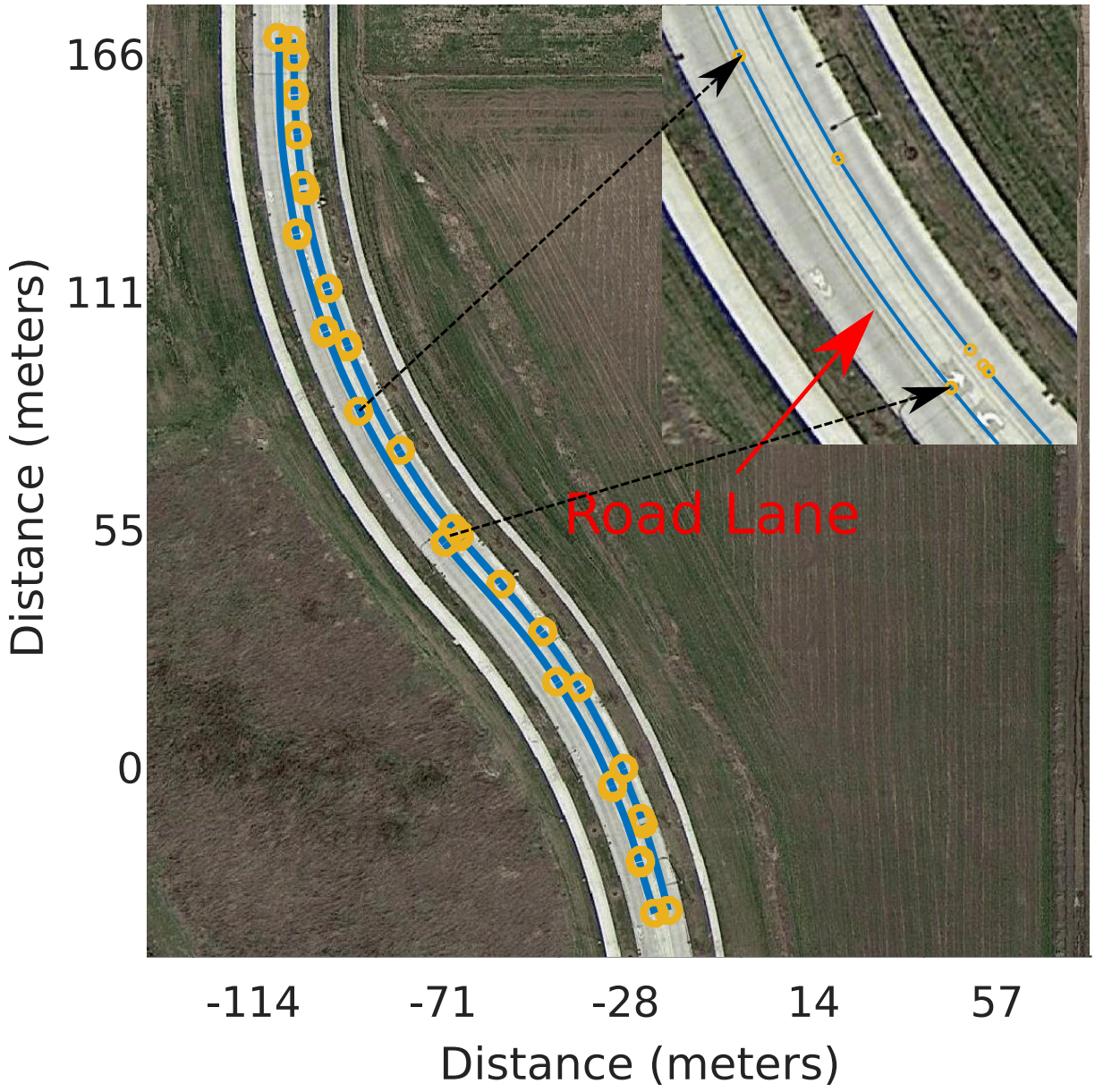


Figure 36: Mapping results plotted in google maps for DS4. The mapping results display the reduced number of curves required to estimate the sidewalk. Yellow points denote the start and end points of curve segments, The blue plot represents the location of curves interpolated with the estimated control points.

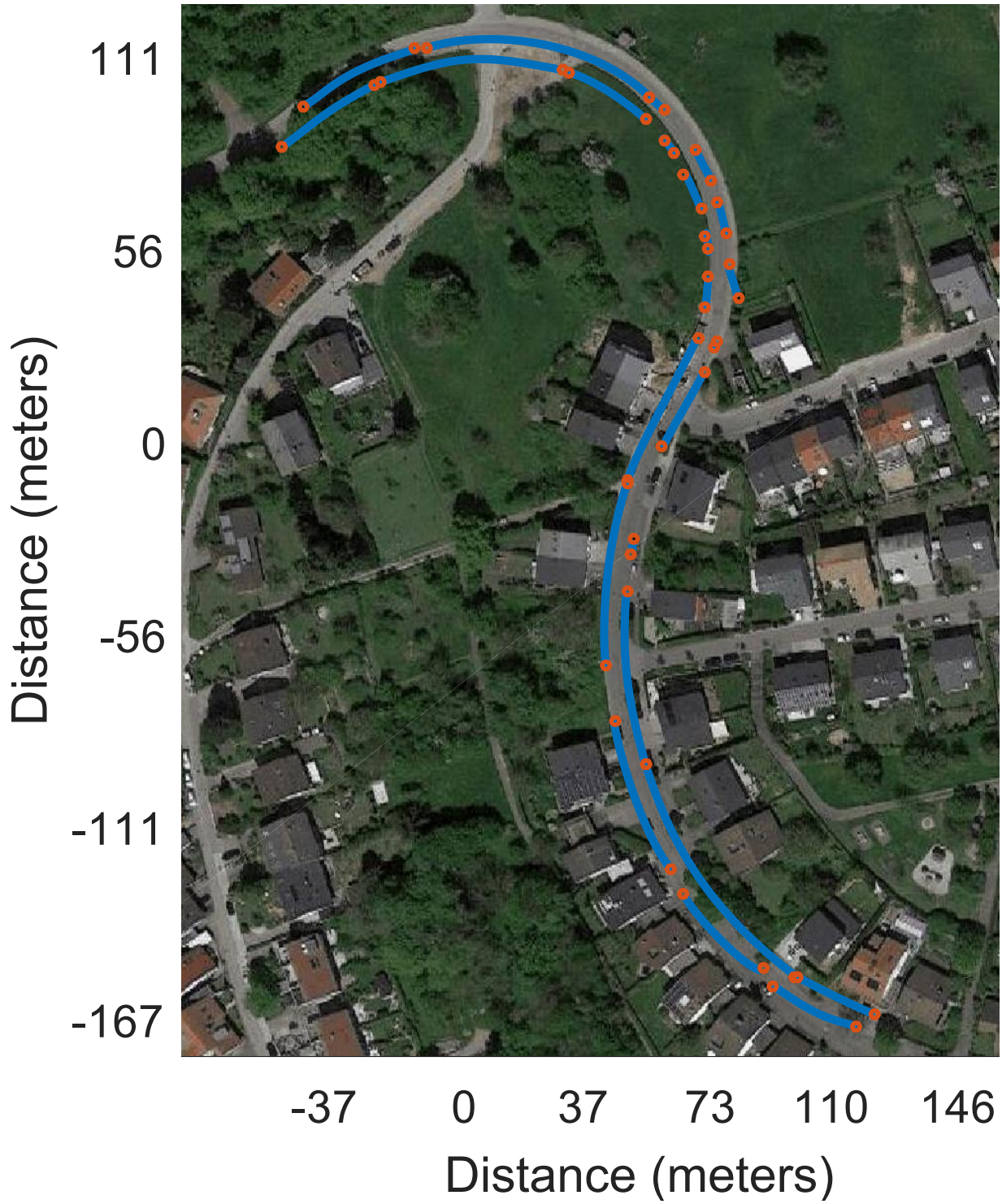


Figure 37: Mapping results plotted in google maps for DS5. The mapping results display the reduced number of curves required to estimate the road. red points denote the start and end points of curve segments, The blue plot represents the location of curves interpolated with the estimated control points.

5.4 Calibration and Parameter Selection

While obtaining the experimental data in this paper, we found the stereo camera to be incredibly sensitive to small disturbances, mostly due to vibrations. Thus, we find it helpful to emphasize that our stereo camera and IMU had a precision mounted case that was specifically created to prevent disturbances from disrupting the calibration. Additionally, stereo images on our sensor platform were time-synchronized within nanoseconds of each other using an external hardware trigger. Furthermore, to enhance the accuracy of the sensor system, the stereo calibration, and the camera to IMU calibration were performed immediately following data collection.

Throughout this paper, we described various parameters. We now describe how to select these parameters. To calculate $\mathbf{V}_r$, we apply the method in [Myers et al., 2010]. To do so, we first estimate the error variance σ^2:

$$\sigma^2 = \frac{\sum_{o\in\{L,R\}} \sum_{j=1}^{m} \sum_{i=1}^{n} ||\mathbf{y}_i - \hat{\mathbf{y}}(\boldsymbol{\beta}))||^2}{d_{\boldsymbol{\beta}}}$$

where $d_{\boldsymbol{\beta}}$ is the number of degrees of freedom for error in the parameter vector $\boldsymbol{\beta}$. An estimate of the parameter covariance $\mathbf{V}_r = \mathrm{Var}(\boldsymbol{\beta})$ is given by

$$\mathbf{V}_r = \sigma^2(\mathbf{J}^\top\mathbf{J})^{-1}$$

where $\mathbf{J}$ is the Jacobian of $\hat{\mathbf{y}}$ with respect to $\boldsymbol{\beta}$, i.e., $\mathbf{J} = \frac{\partial\hat{\mathbf{y}}(\boldsymbol{\beta})}{\partial\boldsymbol{\beta}}$.

The process noise covariance matrix $\mathbf{W}$ is calculated with the method in [Trawny and Roumeliotis, 2005], where the noise characteristics of the IMU are obtained directly from the manufacturer's data sheet: `http:/www.novatel.com/assets/Documents/Papers/SPAN-IGM-A1-PS.pdf`.

The above calculations for $\mathbf{W}$ and $\mathbf{V}_r$ provide a good starting point for tuning $\mathbf{W}$ and $\mathbf{V}_r$. It is important to note that minor hand-tuning of $\mathbf{W}$ and $\mathbf{V}_r$ was required offline to obtain the results in this section. However, this hand-tuning was only required for DS1 and DS5. The filter gains for DS5 are different from the other data sets because DS5 used a completely different sensor platform (DS5 is a sequence taken from the KITTI data set). The filter gains for DS2, DS3, and DS4 are the same as DS1.

For the initial error covariance matrix $\boldsymbol{P}$, we initialize the robot pose block as $\boldsymbol{P}_{r_p,r_p} = \mathbf{0}^{9\times9}$. Likewise, the initial error covariance between a control point and robot pose is initialized to zero, i.e., $\boldsymbol{P}_{r_p,j_p} = \mathbf{0}^{9\times3}$. The error covariance between curve control points are initialized to the initial value of the measurement covariance $\mathbf{V}_0$.

In the initial frame, we obtain a parameter estimate of $\boldsymbol{\beta}$ for use in the Levenberg-Marquardt algorithm in four steps. First, we extract the path boundary. Second, we add curves in the image plane as described in Section 4.2. Third, we match curves between the stereo pair as described in Section 3.2. Fourth, our initial guess for $\boldsymbol{\beta}$ is obtained by triangulating the control points between matched curves.

In Section 3.1, the size ρ_n of the average filter window is $\rho_n = 5 \times 5$. To extract the boundary of the path, the image was processed in the hue, saturation, value (HSV) color space. Due to the stark contrast in hue between the concrete sidewalk and green grass, or the concrete road and yellow road lane (see Figures 12–15), there is a range of valid threshold values. Assuming a normalized image, we set the thresholds to find the sidewalk or yellow road lane to be between the following values: $\mathrm{H_{min}} = 0.09$, $\mathrm{H_{max}} = .5$, $\mathrm{S_{min}} = .15$, $\mathrm{S_{max}} = 1.0$, $\mathrm{V_{min}} = 0$, and $\mathrm{V_{max}} = 1$ for the hue, saturation, and value channels respectively in DS1-DS3. For DS4, we set the thresholds as $\mathrm{H_{min}} = 0.04, \mathrm{H_{max}} = 0.26$, $\mathrm{S_{min}} = .1$, $\mathrm{S_{max}} = 1.0$, $\mathrm{V_{min}} = 0$, and $\mathrm{V_{max}} = 1$. The size ρ_t of the template used to match and refine the location of curves in Section 3.2 is set as $\rho_t = 15\times15$ for the left image. In the right image, the size of the image patch is 17×20. We set ρ_r in Section 3.2 as $\rho_r = 5$ pixels. We set ρ_s in Section 4.1 as $\rho_s = 100$ millimeters for DS1, DS2, DS3, and DS4, and $\rho_s = 2.5$ meters in DS5. To calculate The thresholds for the Mahalanobis distance tests in Section 4.1 are set at 2.5

standard deviations. In Section 4.2, we set the size of the image window ρ_w as $\rho_w = 16 \times 16$. Initially, we set ρ_k to 1.5 pixels. If no corner points are within 1.5 pixels of the path boundary, we progressively increase ρ_k from 2.5 pixels to 3.5 pixels until a corner point is found. If no corner points are in this range, we select the initial boundary point as the desired control point. The Shapiro-Wilk test in Section 4.3 is a parametric hypothesis test of normality. The null hypotheses is that a parameter is normal with unspecified mean and variance. The significance level we implemented for the Shapiro-Wilk test is set at .05. When the path is not sufficiently smooth, the Shapiro-Wilk test had a tendency to over split the boundary. As described in Section 4.3, we incorporated a threshold parameter ρ_p to avoid over splitting the path. While we do not provide a systematic way of selecting ρ_p, we tested a range of values for ρ_p, between two and fifteen pixels, and observed no noticeable effects on the localization accuracy, and very minimal effects on the mapping results. We set ρ_p as $\rho_p = 10$ pixels. We set d in Section 4.8 as $d = 1$ meter.

6 Conclusion

In this paper, we presented a SLAM algorithm that uses Bézier curves as landmark primitives rather than feature points. Our approach allowed us to create an extremely sparse structured map of the environment. We compared our algorithm against SPTAM and OKVIS in five different settings. In the first three environments, a long winding sidewalk provided curve landmarks. In the fourth environment, road lanes provided curve landmarks. In the fifth environment, the road provided curve landmarks. In the first, third, and fourth locations, Curve SLAM was more accurate than SPTAM and OKVIS because it was difficult to track feature points in these environments. In the second environment, SPTAM and OKVIS were slightly more accurate than Curve SLAM. This result is expected because point-based feature detector/extractor tracking algorithms will likely provide a more robust motion estimate than our tracking algorithm when feature points are readily available. In this regard, point-based approaches are not inferior. Indeed, recent point-based SLAM approaches provide precise motion estimates that run in real-time over long trajectories. Curve SLAM can be modified rather easily to include feature points so that curves and feature points can be used simultaneously to solve the SLAM problem. However, alternative approaches are required in order to localize and map settings that lack distinguishable feature points and to provide compact, structured maps of the environment. In all five environments, Curve SLAM required fewer landmarks compared to SPTAM and OKVIS. In fact, in our experimental evaluations, we observed that Curve SLAM was able to reduce the required number of landmark features by several orders of magnitude relative to SPTAM and OKVIS. Future work includes applying our algorithm to a river setting and solving the place recognition problem when the appearance of the environment changes drastically, e.g., at different times of the day, across seasonal changes, or in adverse environmental conditions.

Acknowledgments

The work in this article was supported by the Office of Naval Research (grant number N00014-14-1-0265), and the SMART scholarship for service program (Office of Secretary Defense-Test and Evaluation, Defense-Wide/PE0601120D8Z National Defense Education Program (NDEP)/BA-1, Basic Research)

Appendix

Epipolar Geometry of Curves

Consider a 3-D curve $\mathbf{C}$ that projects to $\mathbf{C}_L$ and $\mathbf{C}_R$ in the left and right stereo images, see Figure 38. From the perspective of the left image, the 3-D location of $\mathbf{C}$ could be located anywhere along the rays that project from the optical center $\mathbf{O}_L$ of the camera passing through the image plane of $\mathbf{C}_L$ to form the points comprising $\mathbf{C}$. With a curve correspondence between the left and right images, the 3-D curve lies at the

intersection of the rays that project from the optical center of each camera. Thus, we can estimate the 3-D location of **C**.

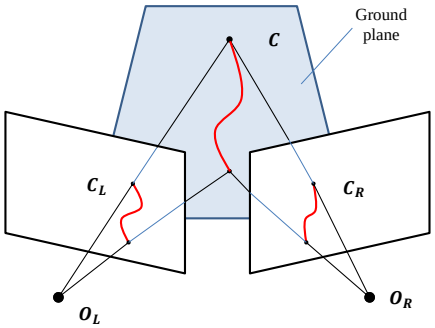


Figure 38: A polynomial curve projected to a stereo image pair

We can prove, under certain assumptions, that the preimage of two image curves is itself a curve in world co-ordinates. This proof is outlined below.

Proof of Curve Reconstruction

For this proof, we make the assumption that for a specific curve in $\mathbb{R}^3$, the map $f : \mathbb{R}^3 \to \mathbb{R}^2$ projecting the curve to an image is an isomorphism. This assumption only implies that the curve is fully observed in both images (i.e., a curve should not lose information and appear as a point or line when projected to the image).

Background:

For a smooth map between manifolds given by $f : X \to Y$, $y \in Y$ is a *regular value* of f if $\forall x \in f^{-1}(y)$, $df_x : T_xX \to T_yY$ is surjective. Here, T_xX and T_yY are the tangent spaces of X and Y at points x and y.

The Preimage Theorem: If $f : X \to Y$ is a smooth map, and $y \in Y$ is a regular value of f, then $M = \{x : x \in f^{-1}(y)\}$ is a submanifold of X, and the codimension of M in X is equal to the dimension of Y.

Proposition 1: Given a stereo image frame, and a curve observed in each image, the preimage is itself a curve in the world frame.

Proof:

We can define a curve in the left image as $f_L(u_L, v_L) = 0$. Under normalized perspective projection, $u_L = x/z$ and $v_L = y/z$. So, we can express this curve as $f_L(x/z, y/z) = 0$, $f_L : \mathbb{R}^3 \to \mathbb{R}^1$.

Then, the inverse image of 0 is given by:

$$M_L = \{(x, y, z) \in \mathbb{R}^3 \mid f_L(x/z, y/z) = 0\}$$

and using the Preimage Theorem, M_L is a manifold in $\mathbb{R}^3$ with codimension 1. Thus, M_L is a 2-manifold, which can be represented in implicit form by the set:

$$M_L = \{(x, y, z) \in \mathbb{R}^3 \mid F_L(x, y, z) = 0\}$$

Similarly, if we define the curve in the right image as $f_R(u_r, v_r) = 0$, using a similar argument the inverse

image of 0 in the right image is also a 2-manifold given by:

$$M_R = \{(x, y, z) \in \mathbb{R}^3 \mid F_R(x, y, z) = 0\}$$

Consider now the function $F : \mathbb{R}^3 \to \mathbb{R}^2$ given by

$$F(x, y, z) = \begin{bmatrix} F_L(x, y, z) \\ F_R(x, y, z) \end{bmatrix}$$

The inverse image M of the stereo image curves is the intersection of the two surfaces M_L and M_R, or the set of points for which $F_L = F_R = 0$:

$$M = \{(x, y, z) \in \mathbb{R}^3 \mid F(x, y, z) = \mathbf{0}]$$

Since in this case, $F : \mathbb{R}^3 \to \mathbb{R}^2$, we can conclude using the Preimage Theorem that the inverse image of the point $[0, 0]^T$ will be a manifold of codimension 2 in $\mathbb{R}^3$ (i.e., a 1-manifold, or a curve).

References

Agarwal, S., Mierle, K., and Others (2015). Ceres Solver. `http://ceres-solver.org`.

Berry, T. G. and Patterson, R. R. (1997). The uniqueness of Bézier control points. *Computer Aided Geometric Design*, 14(9):877–879.

Dani, A., Panahandeh, G., Chung, S.-J., and Hutchinson, S. (2013). Image Moments for Higher-Level Feature Based Navigation. *International Conference on Intelligent Robots and Systems*, pages 602–609. Tokyo, Japan.

Furgale, P. and Barfoot, T. D. (2010). Visual Teach and Repeat for Long-Range Rover Autonomy. *Journal of Field Robotics*, 27(5):534–560.

Furgale, P., Barfoot, T. D., and Sibley, G. (2012). Continuous-Time Batch Estimation using Temporal Basis Functions. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2088–2095. Saint Paul, Minnesota.

Furgale, P., Rehder, J., and Siegwart, R. (2013). Unified and Spatial Calibration for Multi-Sensor Systems. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1280–1286. Tokyo, Japan.

Gee, A., Chekhlov, D., Calway, W., and Mayol-Cuevas, W. (2008). Discovering Higher Level Structure in Visual SLAM. *IEEE Transactions on Robotics*, 24:980–990.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237.

Geiger, A., Lenz, P., and Urtasun, R. (2012). 2012 IEEE Conference on Computer Vision and Pattern Recognition. In *Conference on Computer Vision and PatternRecognition (CVPR)*, pages 3354–3361. Providence, Rhode Island.

Jones, E. S. and Soatto, S. (2011). Visual-inertial navigation, mapping and localization: A scalable real-time causal approach. *The International Journal of Robotics Research*, 30(4):407–430.

Kaess, M., Johannsson, H., Roberts, R., IIa, V., Leonard, J. J., and Dellaert, F. (2012). iSAM2: Incremental smoothing and mapping using the Bayes tree. *International Journal of Robotics Research (IJRR)*, 31(2):216–235.

Kedem, K. and Yarmovski, Y. (1996). Curve based stereo matching using the minimum Hausdorff distance. *In Proeedings of the Twelfth Annual Symposium on Computational Geometry*, pages 415–418.

Keivan, N. and Sibley, G. (2015). Asynchronous adaptive conditioning for visual-inertial SLAM. *International Journal of Robotics Research (IJRR)*, 34(13):1573–1589.

Kelly, J. and Sukhatme, G. S. (2011). Visual-Inertial Sensor Funsion: Localization, Mapping and Sensor-to-Sensor Self-calibration. *The International Journal of Robotics Research*, 30(1):56–79.

Khan, M. (2007). Approximation of data using cubic Bézier curve least square fitting. http://hipp.certec.lth.se/trac/raw-attachment/wiki/BezierPaths/.

Kim, J., Yoon, K.-J., and Kweon, I. S. (2015). Bayesian filter for keyframe-Based visual SLAM. *The International Journal of Robotics Research*, 34(4-5):517–531.

Klein, G. and Murray, D. (2007). Parallel Tracking and Mapping for Small AR Workspaces. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234. Washington D.C.

Konolige, K. and Agrawal, M. (2008). FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping. *IEEE Transactions on Robotics*, 24(5):1066–1077.

Levenberg, K. (1944). A Method for the Solution of Certain Non-Linear Problems in Least Squares. *Quarterly of Applied Mathematics*, 2:164–168.

Li, M. and Mourikis, A. I. (2013). High-precision, consistent EKF-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711.

Linegar, C., Churchill, W., and Newman, P. (2016). Made to Measure: Bespoke Landmarks for 24-Hour, All-Weather Localisation with a Camera. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 787–794, Stockholm, Sweden.

Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P., , and Milford, M. J. (2016). Visual Place Recognition: A Survey. *IEEE Transactions on Robotics*, 32(1):1–19.

Lu, Y. and Song, D. (2015). Visual Navigation Using Heterogeneous Landmarks and Unsupervised Geometric Constraints. *IEEE Transactions on Robotics*, 31(3):736–749.

Lucas, B. D. and Kanade, T. (1981). An Iterative Image Registartion Technique with an Application to Stereo Vision. *Proceedings of the 1981 DARPA Imaging Understanding Workshop*, pages 121–130.

Luetenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2015). Keyframe-based visual-inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334.

Marquardt, D. (1963). An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441.

Maye, J., Furgale, P., and Siegwart, R. (2013). Self-supervised Calibration for Robotic Systems. *IEEE Intelligent Vehicles Symposium*, pages 473–480.

McManus, C., Upcroft, B., and Newman, P. (2015). Learning Place-Dependant Features for Long-Term Vision-Based Localisation. *Autonomous Robots, Special issue on Robotics Science and Systems 2014*, 39(3):363–387.

Meier, K., Chung, S. J., and Hutchinson, S. (2016). Visual-inertial Curve SLAM. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1238–1245. Daejeon, Korea.

Mourikis, A. I. and Roumeliotis, S. I. (2007). A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation. In*: Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3565–3572. Roma, Italy.

Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D. (2015). ORB-SLAM: a Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics*, 31(5):1147–1163.

Myers, R. H., Montgomery, D. C., Vining, G. G., and Robinson, T. J. (2010). *Generalized Linear Models: With Applications in Engineering and the Sciences, Second Edition*. John Wiley & Sons, Inc., Hoboken, NJ, USA.

Nguyen, V., Harati, A., and Siegwart, R. (2007). A Lightweight SLAM algorithm using Orthogonal Planes for Indoor Mobile Robotics. *International Conference on Intelligent Robots and Systems*, pages 658–663.

Nuske, S., Choudhury, S., Jain, S., Chambers, A., Yoder, L., Scherer, S., Chamberlain, L., Cover, H., and Singh, S. (2015). Autonomous Exploration and Motion Planning for an Unmanned Aerial Vehicle Navigating Rivers. *Journal of Field Robotics*, 32(8):1141–1162.

Ozog, P., Carlevaris-Bianco, N., Kim, A., and Eustice, R. M. (2015). Long-term Mapping Techniques for Ship Hull Inspection and Surveillance using an Autonomous Underwater Vehicle. *Journal of Field Robotics*, 33(3):265–289.

Paton, M., Pomerleau, F., MacTavish, K., Ostafew, C. J., and Barfoot, T. D. (2016). Expanding the Limits of Vision-based Localization for Long-term Route-following Autonomy. *Journal of Field Robotics*, doi:10.1002/rob.21669.

Patterson, R. R. (1985). Projective Transformations of the Parameter of a Bernstein-Bézier Curve. *ACM Transactions on Graphics*, 4(4):276–290.

Pedraza, L., Rodriguez-Losada, D., Matia, F., Dissanayake, G., and Miro, J. V. (2009). Extending the Limits of Feature-Based SLAM With B-Splines. *IEEE Transactions on Robotics*, 25(2):353–366.

Piegl, L. and Tiller, W. (1997). *The NURBS Book, Second Edition*. Springer-Verlag New York, INC. New York, NY, USA.

Pire, T., Fischer, T., Civera, J., Cristóforis, P. D., and Berlles, J. J. (2015). Stereo Parallel Tracking and Mapping for Robot Localization. *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, pages 1373–1378. Hamburg, Germany.

Rao, D., Chung, S.-J., and Hutchinson, S. (2012). Curve SLAM: An approach for Vision-based Navigation without Point Features. *2012 IEEE/RSJ International Conterence on Intelligent Robots and Systems*, pages 4198 – 4204.

Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H. J., and Davison, A. J. (2013). SLAM++: Simultaneous Localisation and Mapping at the Level of Objects. *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1352–1359. Portland, Oregon.

Salomon, D. (2006). *Curves and Surfaces for Computer Graphics*. Springer Science+Business Media, Inc., New York.

Schmid, C. and Zisserman, A. (2000). The Geometry and Matching of Lines and Curves Over Multiple Views. *International Journal of Computer Vision*, 40(3):199–233.

Shapiro, S. and Wilk, M. (1965). An Analysis of Variance Test for Normality. *Biometrika*, 52(3/4):591–611.

Shi, J. and Tomasi, C. (1994). Good Features to Track. *9th IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600. Seattle, WA.

Smith, P., Reid, I., and Davison, A. (2006). Real-Time Monocular SLAM with Straight Lines. *Proceedings-British Machine Vision Confererence*, pages 17–26.

Suzuki, S. and Abe, K. (1985). Topological Structural Analysis of Digitized Binary Image by Border Following. *Computer Vision Graphics and Image Processing*, 30(1):32–46.

Teichmann, M., Weber, M., Zoellner, M., Cipolla, R., and Urtasun, R. (2016). MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving. *arXiv preprint arXiv:1612.07695*.

Trawny, N. and Roumeliotis, S. I. (2005). Indirect Kalman Filter for 3D Attitude Estimation A Tutorial for Quaternion Algebra. Technical Report 2005-002, Rev. 57, MARS Lab, University of Minnesota.

Walter, M. and Fournier, A. (1996). Approximate arc length parameterization. In *Proceedings of the 9th Brazilian Symposium on Computer Graphics and Image Processing*, pages 143–150.

Wang, H., Kearney, J., and Atkinson, K. (2002). Arc-length Paramaterized Spline Curve for Real-time Simulation. In *Proc. 5th International Conference on Curves and Surfaces*, pages 387–396.

Xiao, Y. and Li, Y. (2005). Optimized Stereo Reconstruction of Free-form Space Curves Based on a Nonuniform Rational B-spline Model. *Journal of the Optical Society of America*, 22(9):1746–62.

Yang, J., Dani, A., Chung, S.-J., and Hutchinson, S. (2015). Vision-Based Localization and Robot-Centric Mapping in Riverine Environments. *Journal of Field Robotics*, 34(3):429–450.

Zhang, G., Lee, J. H., Lim, J., and Suh, I. H. (2015). Building a 3-D Line-Based Map Using Stereo SLAM. *IEEE Transactions on Robotics*, 31(6):1364–1377.